Departamento de Ciencias da Computación e
Tecnoloxías da Información
Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑERÍA INFORMÁTICA
MENCIÓN EN SISTEMAS DE INFORMACIÓN

Sistema de información para la gestión de rutas y diario de vuelo

Estudiante: Pablo Estévez Álvarez
Dirección: Alejandro Cortiñas Álvarez
Miguel Ángel Rodríguez Luaces

A Coruña, 12 de febrero de 2020

A Marta y Chema, por agrandarme las alas todos los días.

Agradecementos

En primer lugar me parece importante agradecer a Álex y Miguel el trabajo que han hecho para que el proyecto salga adelante, han estado pendientes de mí siempre que lo he necesitado.

También a todos los compañeros con los que he compartido clases en la carrera, en especial a Víctor y Jaime, sin ellos no sabría todo lo que sé.

A mi familia y mis amigos, porque son una parte fundamental de mi vida y siempre me están apoyando.

Y a Vane, porque me acompaña en las buenas y en las malas, sin importar los motivos ni los resultados.

Resumen

El objetivo de este trabajo de fin de grado es desarrollar una aplicación destinada a pilotos y usuarios de aeronaves en la que podrán crear de rutas de vuelo y publicarlas para que el resto de usuarios puedan visualizarlas, comentarlas y descargarlas. Además se incorporará un apartado para que los pilotos puedan rellenar su diario de vuelo y guardar las entradas en una tabla, desde la que se podrá ver en detalle cada vuelo.

Para ello se tendrán en cuenta tres tipos de usuario, un administrador que gestionará la creación de aeródromos y aeronaves, además de moderar los comentarios de las rutas, un usuario anónimo, el cual podrá visualizar una lista de las rutas publicas y sus detalles, y un usuario registrado que podrá crear rutas y entradas de su diario de vuelo, además de visualizar listas y detalles de las rutas publicadas por otros usuarios para descargarlas y comentarlas.

Para alcanzar los objetivos de este trabajo de fin de grado ha sido necesario realizar un análisis previo para profundizar en los objetivos del proyecto y analizar el dominio con el que se iba a trabajar. A continuación, se llevó a cabo el desarrollo y las pruebas de las distintas funcionalidades de la aplicación.

Para su desarrollo se ha creado un servidor con Java y Spring para albergar la lógica de la aplicación. Este servidor implementa un servicio REST para facilitar la comunicación con el cliente WEB basado en React, el cual ofrecerá al usuario una interfaz con la que interactuar para llevar a cabo las funcionalidades disponibles. Además, un SGBD relacional creado con PostgreSQL facilitará el almacenamiento de la información.

El trabajo de fin de grado se gestionó siguiendo una metodología iterativa e incremental para la creación de software basada en conceptos y directrices de Scrum que dividió el proceso en ocho iteraciones, las cuales comenzaron con una reunión de revisión de la iteración anterior y planificación de la siguiente. En cada una de estas iteraciones se llevó a cabo el análisis, diseño e implementación de las funcionalidades asignadas a ella.

Abstract

The objective of this end-of-degree project is to develop an application intended to pilots and users of aircraft where they will be able to upload flight routes for the rest of the users to view, comment and download. Also, the application will have a section where pilots can fill their logbook and save every entry in a table where they can check flight details.

For this purpose three types of user will be considered. An administrator will manage creation of aircraft and aerodromes as well as moderation of comments in flight routes, an anonymous user that will see a route list and its details, and finally a registered user that will

be able to upload flight routes and fill its pilot logbook, as well as check route lists and route details for him to comment and download.

In order to achieve this goals, it was necessary to perform a previous analysis to go deeper into the objectives of the project and analyse the domain. Next, every application functionality was implemented and tested.

In the development, a server was created with Java and Spring to contain the logic of the application. This server implements a REST service to ease communication with a React WEB client, which offers an interface for the user to interact with it and utilize all functionalities available. Also, a PostgreSQL DBMS will ensure data storage.

The end-of-degree work was managed following an iterative and incremental methodology for software development based on the guidelines of Scrum. The process was divided in eight iterations that started with a meeting analyzing last iteration and planning next. Every one of this iterations was destined to analyse, design and implement all the functionalities that were assigned to it.

Palabras chave:

- Java
- Spring
- Hibernate
- Servicio REST
- Aplicación Web
- PostgreSQL
- Javascript
- React
- Redux
- Leaflet
- Piloto
- Diario de vuelo
- Ruta de vuelo

Keywords:

- Java
- Spring
- Hibernate
- REST service
- Web Application
- PostgreSQL
- Javascript
- React
- Redux
- Leaflet
- Pilot
- Pilot logbook
- Flight route

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1 Motivación

El sector de la aviación, a pesar de contar solamente con poco más de cien años de historia, está experimentando un crecimiento exponencial, así como la tecnología asociada a él. Este crecimiento provoca que cada vez haya más gente allegada al sector, ya sea como medio de transporte o como actividad de recreo. Por eso, en la era de las redes sociales, comienza a surgir la necesidad de conectar a todas esas personas. Existen aplicaciones en las que se puede crear rutas, principalmente de senderismo, y compartirlas con el resto de usuarios para que estos las puedan guardar y descargar. A partir de esta idea, sería interesante tener la posibilidad, ya sea como piloto o como usuario de aeronaves, de poder hacer lo mismo con rutas de vuelo.

Por otro lado, los pilotos de aeronaves disponen de un diario de vuelo en el que deben anotar los vuelos realizados y sus características. Aunque existen aplicaciones para rellenarlo en línea, muchos pilotos no encuentran la motivación para dejar de hacerlo a mano, lo que invita al desarrollador a crear una aplicación sencilla en la que los usuarios, además de poder rellenar su diario de vuelo, tengan algún tipo de incentivo para que les merezca la pena dejar de hacerlo a mano.

Por todo eso, el objetivo principal de este proyecto será la creación de una aplicación Web con una interfaz agradable y sencilla que permita la creación de entradas del diario de vuelo y su posterior visualización en una tabla en la cual se concentren todas las entradas para ordenarlas como quiera el piloto. Además, la aplicación añade el incentivo de que se les permita guardar sus rutas y compartirlas con el resto de usuarios para que puedan visualizarlas y comentarlas, además de descargarlas en sus GPS.

1.2 Objetivos

Para poder alcanzar el objetivo principal de este trabajo de fin de grado descrito en la sección anterior, se han planteado una serie de objetivos específicos:

- Interesa que la aplicación llegue a cualquier tipo de público, por lo que dispondrá de **funcionalidades para usuarios anónimos**, que estarán limitadas a visualizar el contenido publicado por los pilotos registrados, con ciertas restricciones para incentivar su registro.
- Proporcionar un sistema de **registro de usuarios** para que los pilotos puedan disfrutar de una experiencia completa de la aplicación, permitiéndoles utilizar las funciones que no estarán disponibles para usuarios anónimos.
- Los usuarios registrados dispondrán de un **perfil de usuario** en el que se reflejará su información personal, así como una lista de sus rutas publicadas, estadísticas de vuelo y lista de seguidores. Además un piloto podrá visitar el perfil de otro para poder añadirlo a su lista de pilotos seguidos.
- Un piloto autenticado podrá crear **rutas de vuelo** y publicarlas. Estas rutas aparecerán en tres listas: una para las rutas creadas por el usuario, otra para las publicadas por el resto de usuarios y una última para las que haya marcado como favoritas. La creación y modificación de las rutas se realizará a través de formularios sencillos. Además, cada ruta dispondrá de una vista de detalle.
- Otra de las grandes ventajas de la aplicación será la posibilidad de **rellenar el diario de vuelo del piloto** y su posterior visualización en una tabla. Esta funcionalidad también debe disponer de un formulario sencillo e intuitivo. Además, cada entrada de la tabla dispondrá de una vista de detalle para que el piloto revise con exactitud cada campo del diario.
- Al disponer de un registro de usuarios en el que cualquiera se puede crear un perfil en la aplicación, se cree necesaria una figura que modere los comentarios en las rutas publicadas. Para ello, la aplicación contará con una serie de **funcionalidades de administración**, que, además de labores de moderación, permitirán la creación y modificación de aeródromos y aeronaves.
- Asegurar un **producto de calidad**, intentando que que la experiencia de uso sea lo más placentera posible para el usuario, minimizando la aparición de errores. También se debe asegurar la privacidad de las rutas que los pilotos no publiquen y sus diarios de vuelo.

Capítulo 2

Fundamentos tecnológicos

2.1 Estado del arte

A partir de los objetivos descritos en la sección anterior, se han buscado aplicaciones existentes que den soporte a las necesidades expresadas para tener una perspectiva inicial. Se han encontrado dos aplicaciones que destacan: Wikiloc y FlyLog.io.

Wikiloc [1] es una aplicación web que permite crear y publicar rutas al aire libre referenciadas por GPS para compartirlas con el resto de usuarios y que éstos las puedan ver en detalle y descargar. Aunque se pueden subir rutas de todo tipo, se utiliza principalmente para senderismo y ciclismo.

Figura 2.1: Logo de Wikiloc

Esta herramienta facilita la creación de rutas, ya sea subiéndola directamente desde un archivo importado de un GPS o indicando mano en un mapa los puntos que la forman, para su posterior visualización en detalle, mostrando sus características principales, así como una descripción e imágenes subidas por el creador. También se tiene la posibilidad de descargar un archivo que contiene la información geográfica de la ruta para importarlo en cualquier GPS y poder seguirla en cualquier momento.

Además de esto, cada usuario dispone de su propio perfil y tendrá la posibilidad de seguir a otros usuarios para que el sistema le notifique cuando estos suban una ruta.

Aunque todas estas funciones son gratuitas, la aplicación dispone de una suscripción para tener acceso a otras exclusivas como el seguimiento en vivo de un usuario.

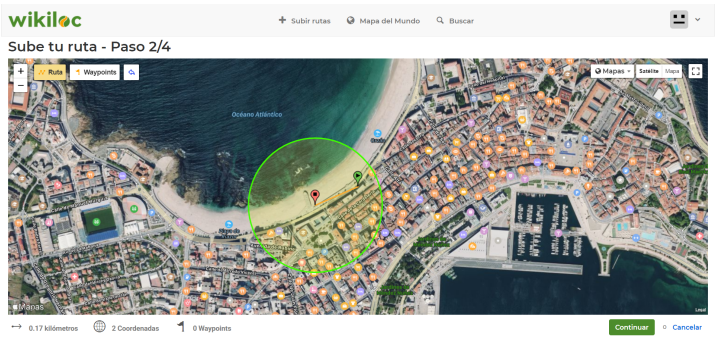


Figura 2.2: Pantalla de creación de una ruta

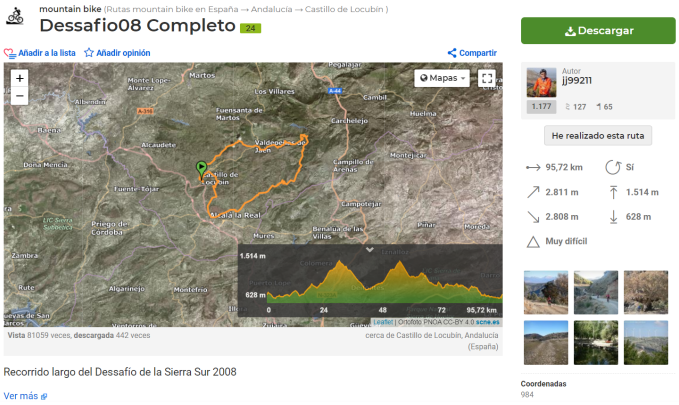


Figura 2.3: Pantalla de detalle de una ruta

En el caso de **FlyLog.io** [2] nos encontramos ante una herramienta cuya principal funcionalidad es crear un diario de vuelo y añadir entradas al mismo, las cuales se pueden consultar, filtrar y descargar en el formato permitido por las distintas administraciones de aviación mundiales.

En la pantalla de lista de vuelos se muestra la información imprescindible de cada entrada, la cuál se puede ver con más exactitud haciendo clicando en el vuelo.

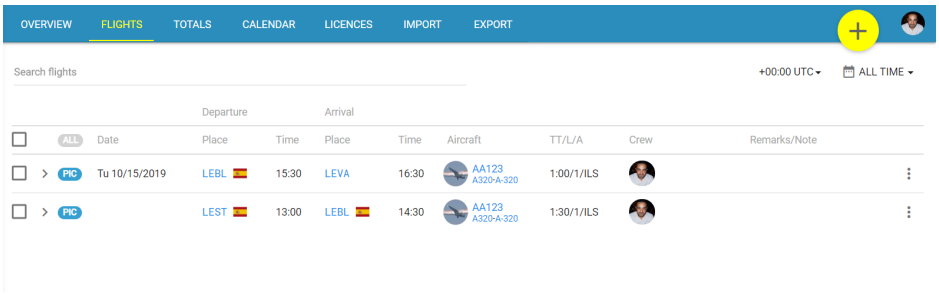


Figura 2.4: Pantalla con la lista de vuelos

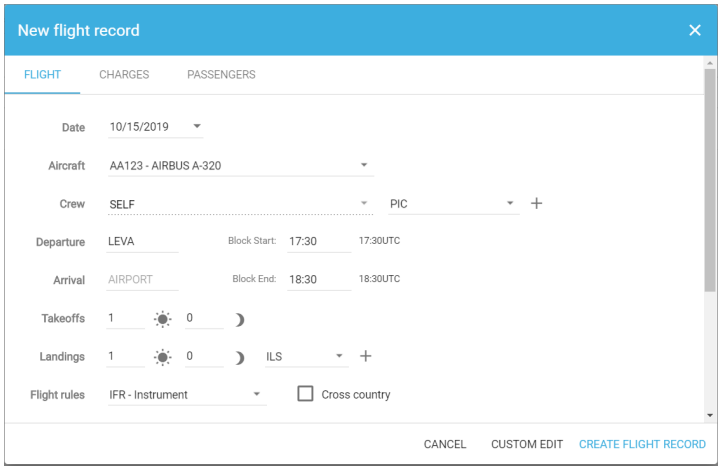


Figura 2.5: Pantalla de creación de un vuelo

Una vez analizadas las características de estas dos aplicaciones, se ha determinado que el objetivo principal de este proyecto pasa por tomar como referencia las funciones principales de éstas y unirlas en una sola herramienta adaptada a usuarios de aeronaves, ya que ellas por sí solas no soportan todas las funcionalidades pretendidas para este trabajo. De **Wikiloc** se intentará reproducir la sencillez de la disposición a la hora de mostrar los detalles de las rutas, y de **FlyLog.io** la facilidad de crear una entrada del diario de vuelo y mostrarla después en una tabla con la información más importante.

2.2 Tecnologías utilizadas

En esta sección se explicarán brevemente las tecnologías que se han utilizado en el desarrollo de este trabajo:

- **Hibernate.** *Framework ORM (Object-Relational Mapping)* que facilita el mapeo de atributos entre la base de datos y el modelo de datos de la aplicación [3].
- **Hibernate Spatial.** Estandariza los datos geográficos para facilitar su manejo [4].
- **Spring.** *Framework* que facilita el desarrollo de aplicaciones Java [5].
- **PostgreSQL.** Sistema de gestión de bases de datos relacional [6].
- **PostGIS.** Extensión de PostgreSQL para trabajar con tipos de datos espaciales [7].
- **Node.js.** plataforma de desarrollo para la creación de aplicaciones destinadas a la web, orientada a redes y centrada en la velocidad y la escalabilidad [8].

- **JWT (JSON Web Tokens).** Estándar para transmitir información de usuario de forma segura entre un cliente y un servidor [9].
- **HTML (HiperText Markup Language).** Lenguaje de marcado para describir la estructura de una página web.
- **CSS (Cascading StyleSheets).** Lenguaje de diseño gráfico para páginas web. Se emplea junto a HTML aunque son independientes.
- **Bootstrap.** Conjunto de herramientas de código abierto para diseño de aplicaciones web [10].
- **JavaScript.** Lenguaje de programación utilizado para crear páginas web dinámicas.
- **React.** Librería JavaScript de código abierto diseñada para la creación de interfaces de usuario [11].
- **Redux.** Librería JavaScript de código abierto para el manejo del estado de una aplicación [12].
- **Axios.** Cliente HTTP basado en promesas para enviar peticiones de forma asíncrona a un servicio REST [13].
- **Leaflet.** Librería JavaScript para crear aplicaciones con mapas interactivos [14].
- **GPX.** Esquema XML para transferir datos GPS entre aplicaciones [15].

Capítulo 3

Metodología y planificación

3.1 Metodología de desarrollo

Para el desarrollo de este proyecto se ha escogido una metodología incremental, que se caracteriza por dividir el ciclo de vida del proyecto en iteraciones que se suceden en el tiempo. Cada una de estas iteraciones supone un incremento en la funcionalidad hasta alcanzar un producto entregable.

En este tipo de metodología se le permite al desarrollador sacar ventaja de lo aprendido de la iteración anterior y ofrece una rápida adaptación a los cambios. Con esto se pueden producir de forma rápida y continua incrementos del producto para poder ofrecérselos al cliente y así obtener *feedback* del mismo para detectar errores rápidamente.

Las iteraciones comienzan planificando el trabajo a realizar teniendo en cuenta la experiencia en las iteraciones previas, ya que de este modo será más sencillo estimar el tiempo necesario.

Para su puesta en práctica se ha decidido adaptar **Scrum** [16], un marco de trabajo para el desarrollo y mantenimiento de productos complejos. Debido a que este marco de trabajo está concebido para equipos de entre cinco y nueve personas, y en este trabajo participan solamente tres, no ha sido posible seguir los principios básicos que éste ofrece, sin embargo, se han adaptado sus directrices a las características del proyecto.

A continuación se muestran los principales conceptos de Scrum y cómo han tenido que ser adaptados.

3.1.1 Equipo Scrum

Grupo autoorganizado y multifuncional que no depende de personas externas para llevar a cabo el trabajo. En él se distinguen tres roles:

- **Dueño del Producto *(Product Owner)***. Es el responsable de maximizar el valor del

producto y el trabajo del Equipo de Desarrollo. Es el único responsable de gestionar la Pila del Producto. Debe escoger los elementos involucrados en cada iteración, ordenarlos para alcanzar los objetivos de manera eficiente y asegurarse de que el Equipo de Desarrollo la entiende.

- **Equipo de Desarrollo *(Development Team)***. Consiste en las personas encargadas de entregar los incrementos del producto a partir de los requisitos establecidos por el *Product Owner*.

- ***Scrum Master***. Responsable de que Scrum es entendido y de que el Equipo Scrum trabaja ajustándose a la teoría, reglas y prácticas del método.

Para este proyecto se ha prescindido de la figura del *Scrum Master* y el Equipo de Desarrollo estuvo formado por un único miembro, el autor del trabajo. El autor y los directores han ejercido a la vez de *Product Owner*, tomando de forma conjunta las decisiones acerca de la priorización de los objetivos de las iteraciones.

3.1.2 Eventos de Scrum

Eventos predefinidos con una duración máxima que tienen como objetivo crear regularidad y minimizar la necesidad de reuniones no definidas en Scrum.

- ***Sprint***. Bloque de tiempo, de tres semanas en el caso de este proyecto, durante el cual se crea un incremento del producto.

- **Reunión de planificación del *Sprint (Sprint Planning Meeting)***. Espacio de tiempo en el que se planifica el trabajo a realizar durante el *Sprint*.

- **Scrum Diario *(Daily Scrum)***. Reunión de corta duración para que el Equipo de Desarrollo sincronice sus actividades y planifique el trabajo para el día siguiente.

- **Revisión del *Sprint (Sprint Review)***. Reunión para revisar el incremento y la Pila del Producto si fuera necesario. En ella los componentes del Equipo Scrum colaboran acerca de lo que se ha hecho durante el *Sprint* y lo ponen en común para tener información útil para las próximas iteraciones. Como resultado de este encuentro se tiene una Pila de Producto revisada.

- **Retrospectiva de *Sprint (Sprint Retrospective)***. Tiene lugar después de la Revisión del Sprint y es una oportunidad para que el Equipo de Desarrollo se inspeccione a sí mismo y cree un plan de mejoras para la siguiente iteración.

En el caso de este proyecto, las reuniones de planificación y revisión del *Sprint* se han llevado a cabo al mismo tiempo, una vez finalizado cada *Sprint*. El Scrum Diario y la Retrospectiva no han tenido lugar en ningún momento al constar el Equipo de Desarrollo de un solo miembro.

3.1.3 Artefactos de Scrum

Son los elementos que forman parte del ciclo de vida de Scrum. Representan trabajo o valor en diversas formas que son útiles para proporcionar transparencia y oportunidades para la inspección y adaptación.

- **Pila de Producto *(Product Backlog)***. Lista ordenada de todo lo que podría ser necesario incluir en el producto. Es la única fuente de requisitos para cualquier cambio a realizarse en el producto. El *Product Owner* es el responsable de su contenido, disponibilidad y ordenación.
- **Pila de Pendientes del *Sprint (Sprint Backlog)***. Conjunto de elementos del *Product Backlog* seleccionados para formar parte de un *Sprint*.

Estos elementos forman parte de este proyecto, estando detallados en la Sección 3.2.2.

3.1.4 Herramientas de apoyo a la metodología

Se han utilizado las siguientes herramientas para facilitar las distintas tareas relacionadas con el proyecto:

- **Eclipse.** IDE utilizado para el desarrollo en Java de la lógica de la aplicación [17].
- **Visual Studio Code.** Entorno de desarrollo utilizado para la creación del cliente con React [18].
- **Google Chrome.** Navegador web utilizado para ejecutar el proyecto.
- **MagicDraw.** Herramienta CASE compatible con el estándar UML para crear diagramas de clase [19].
- **Balsamiq Mockups.** Herramienta que permite diseñar de forma sencilla maquetas de interfaces web [20].
- **Draw.io.** Aplicación web para crear diagramas dentro del navegador [21].
- **Npm.** Gestor de paquetes para JavaScript que facilita la inclusión de librerías en la parte del cliente [22].
- **Overleaf.** Editor de *LaTex* online utilizado para la creación de esta memoria [23].

3.2 Planificación y seguimiento

En esta sección se describirán las tareas necesarias para la realización del proyecto y los recursos utilizados para estimar el coste del trabajo. También se tendrán en cuenta las desviaciones producidas y el motivo de las mismas.

3.2.1 Planificación inicial

Antes de comenzar el desarrollo de la aplicación se realizó un trabajo de análisis para sentar las bases del proyecto y para tener una guía que seguir en el momento del desarrollo. Para ello se han seguido los siguientes pasos:

- En primer lugar se diseñó un **modelo de datos** para comprender los tipos de datos que contendría la base de datos y su relación entre ellos para poder establecer las condiciones necesarias para que reflejen la realidad pretendida.
- Una vez diseñado el modelo de datos, se creó una **pila inicial de producto** con una serie de historias de usuario para tener una referencia a la hora de escoger las funcionalidades a realizar en cada *Sprint*. Esta pila de producto ha ido variando a lo largo del proyecto con los cambios que se han considerado necesarios en las distintas reuniones llevadas a cabo por el Equipo Scrum.
- Teniendo en cuenta las historias de usuario, se consideró importante crear **prototipos de la interfaz de usuario** de la aplicación para tener una idea inicial de las pantallas necesarias para cubrir todas las historias de la pila de producto y los distintos casos de uso presentes en cada una de ellas.
- Por último, un **estudio de las tecnologías** necesarias para el desarrollo del proyecto para conocer las principales características de cada una y decidir qué alternativas se ajustan en mayor medida a los requerimientos del trabajo.

Para finalizar el análisis inicial, se ha creado el diagrama de Gantt de la Figura 3.1a que muestra la planificación inicial del proyecto. Al tener una fecha límite fija, se ha dividido el espacio de tiempo disponible teniendo en cuenta que el análisis preliminar y la memoria final se llevarían a cabo en cuatro semanas respectivamente y que el tiempo restante lo ocuparían los *Sprints* o iteraciones de desarrollo de tres semanas cada una, lo que ha resultado en un total de seis *Sprints*. La decisión de que estas iteraciones durarían tres semanas se tomó en base a que *Sprints* más largos implicarían más material que evaluar en las reuniones de revisión, lo que podría provocar que el trabajo no fuera evaluado de forma correcta, mientras que si los acortamos, se celebrarían reuniones demasiado cortas ya que no se dispondría de suficiente contenido para revisar.

En cuanto a los **recursos** necesarios para llevar a cabo este proyecto, se distinguen dos categorías:

- Por una parte los **recursos humanos**, que incluyen un equipo de tres personas formado por los directores y el autor del proyecto. En el caso de los primeros, ejercerán tareas de planificación y revisión del desarrollo del trabajo, con un coste por hora de **26 €** cada uno teniendo en cuenta que el salario anual de un director de proyecto será de unos 35000 €, sumándole el 32% del coste de la seguridad social en un año de 1770 horas laborables. El autor ha ejercido labores de análisis, diseño y desarrollo de la aplicación, teniendo un coste por hora de **19 €**, teniendo un salario de 25000 € y aplicando los mismos criterios que a un director de proyecto.

- Los **recursos técnicos** empleados incluyen las herramientas software descritas en la Sección 3.1.4, que no tienen ningún tipo de coste al tratarse de soluciones abiertas o versiones para estudiantes, además de un ordenador portátil personal en el que se llevó a cabo el desarrollo del proyecto.

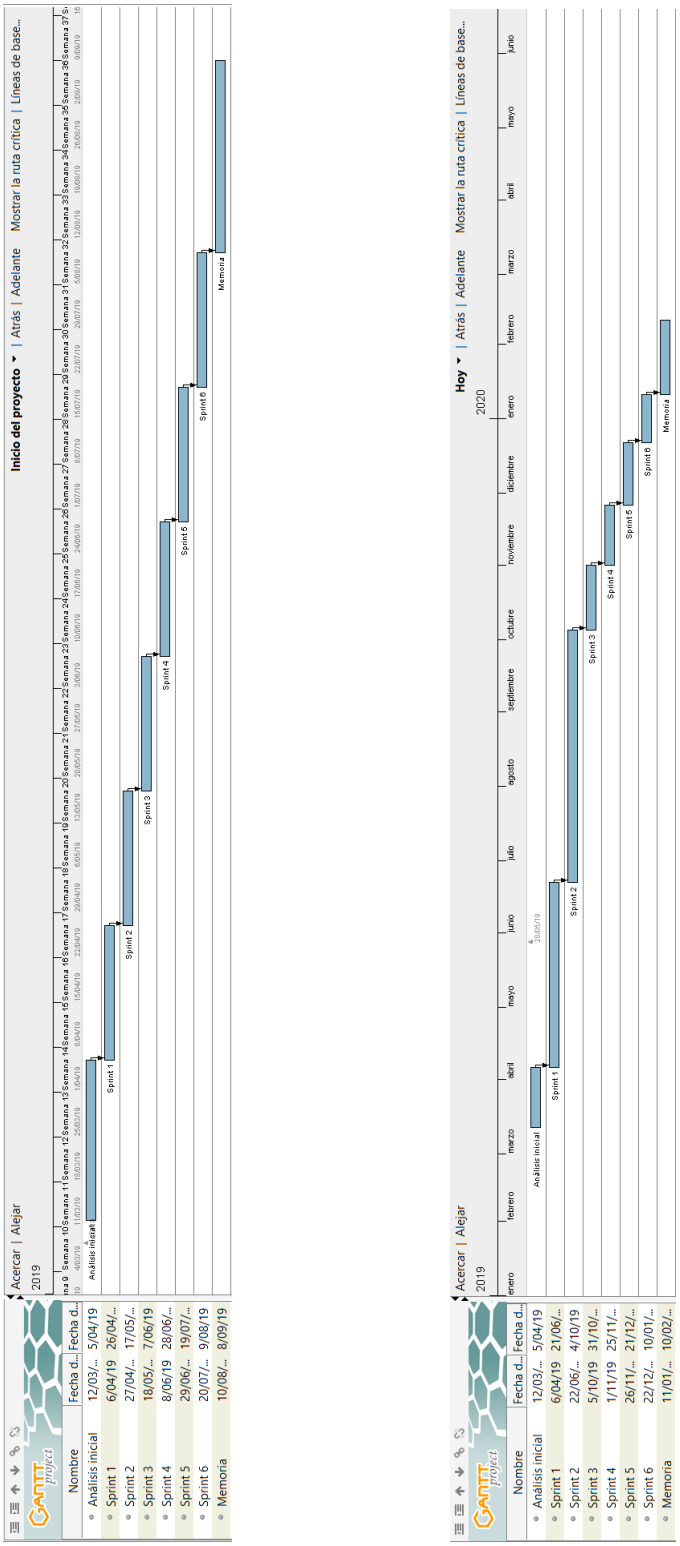
Se ha tenido en cuenta que el autor del proyecto trabajará **4 horas** al día, todos los días de la semana, lo que supone una aproximación de **28 horas semanales**. Conociendo que el proyecto tendrá una duración aproximada de **180 días**, se puede estimar las horas totales de trabajo del autor en **720**. En cuanto a los directores, se han estimado 35 horas totales entre las reuniones celebradas y la revisión de la memoria. Estos **costes** se ven reflejados en el Cuadro 3.1.

Miembro	Horas como director	Horas como desarrollador	Coste por hora	Coste total
Miguel Luaces	35	-	26 €	273 €
Alejandro Cortiñas	35	-	26 €	273 €
Pablo Estévez	-	720	19 €	13680 €
			Total	**14226 €**

Cuadro 3.1: Tabla de costes

3.2.2 Seguimiento de la planificación

Se hablará ahora de cada *Sprint* de forma individual, el trabajo que se llevó a cabo en cada uno de ellos y las desviaciones producidas sobre la planificación inicial. En la Figura 3.1b se puede ver un diagrama de Gantt con la duración de todas las tareas una vez producidas las desviaciones explicadas a continuación.



(a) Diagrama de Gantt de la planificación (b) Diagrama de Gantt del seguimiento

Figura 3.1: Comparación entre planificación y seguimiento

- **Sprint 1.** En esta primera iteración se creó el esqueleto del cliente Web. Además, se incluyó un sistema de autenticación y registro de usuarios y la opción para los usuarios registrados de recuperar su contraseña en caso de haberla olvidado. También se creó el módulo de gestión de aeronaves del administrador. Al tener el autor desconocimiento de las tecnologías utilizadas para el desarrollo del cliente, hubo problemas a la hora de crear el esqueleto inicial y al controlar el estado de la página. Por eso, este primer *sprint* duró ocho semanas más de lo previsto en la planificación inicial.

- **Sprint 2.** A continuación se añadieron las pantallas de gestión de aeródromos en la parte del administrador y se creó un perfil de usuario para los pilotos que muestra su información personal. En esta iteración se conoció que sería imposible entregar el trabajo en la fecha establecida por motivos académicos, por lo que se propuso una nueva fecha de entrega. Al disponer de más tiempo y teniendo en cuenta los problemas surgidos en la primera iteración, se sopesaron las opciones de cambiar de tecnología para el cliente o de alargar el *Sprint* para profundizar en el conocimiento de la actual. Finalmente se optó por la segunda opción, por lo que el *Sprint* se alargó tres meses para estudiar con mas detenimiento los fundamentos de dicha tecnología.

- **Sprint 3.** En el tercer *Sprint* se creó una de las características principales de la aplicación: el diario de vuelo. Además de esto, se añadió al perfil de usuario una pestaña con estadísticas de los vuelos y listas de seguidores. A partir de esta iteración, los *Sprints* tendrán la duración acordada en la planificación inicial y no se producirán más desviaciones.

- **Sprint 4.** Una vez creado el diario de vuelo se comienza a trabajar en la otra gran característica de la aplicación: la creación de rutas. En esta iteración se desarrollaron los formularios de creación y edición de las mismas, además de los algoritmos necesarios para la importación del archivo GPS, los cuáles se explican en la Sección 6.2.1.

- **Sprint 5.** En este *sprint* se crean las diferentes listas que contendrán las rutas. En la pantalla de rutas se encontrarán tres listas: rutas públicas, rutas creadas y rutas favoritas, y en el perfil de usuario se añadirá una pestaña con sus rutas publicadas. También se comienza a trabajar en el detalle de la ruta, añadiendo el mapa, las imágenes adjuntas y la descripción.

- **Sprint 6.** En esta última iteración se termina de crear el detalle de ruta, añadiendo detalles de las imágenes y se añade también la posibilidad de comentarlas. Por último, se crean las páginas principales que se les mostrará a los tres tipos de actores: administradores, usuarios registrados y usuarios anónimos.

Capítulo 4

Análisis

4.1 Requisitos

4.1.1 Actores

Se han distinguido tres entidades básicas que interactuarán con la aplicación:

- **Usuario anónimo.** Este actor solamente tendrá acceso a la lista de rutas públicas y sus detalles y a los perfiles de los usuarios registrados. Podrá registrarse o iniciar sesión en caso de que tenga una cuenta previamente creada.
- **Usuario registrado.** Un usuario que haya creado una cuenta tendrá su propio perfil y podrá seguir a otros usuarios para mantener una lista con seguidores y otra con usuarios a los que sigue. Además se le permitirá crear sus propias rutas, comentar las rutas de los demás pilotos y tener su propio diario de vuelo, permitiéndosele crear entradas, verlas juntas en una tabla y consultar los detalles de cada una.
- **Administrador.** Será el encargado de gestionar la creación y edición de aeródromos y aeronaves. Además tendrá la responsabilidad de moderar los comentarios de las rutas.

4.1.2 Requisitos funcionales

Los requisitos fundamentales que se acordaron en el análisis preliminar de la aplicación fueron los siguientes:

- El sistema debe permitir la **autenticación de usuarios** para que un usuario anónimo se registre y poder iniciar sesión como usuario registrado.
- El administrador tendrá la posibilidad de **gestionar la creación y edición de aeródromos y aeronaves**, para ello se necesitarán formularios específicos para cada una de las entidades y listas para visualizarlas.

- El administrador deberá también **moderar comentarios** en las rutas públicas.
- Un usuario registrado podrá **crear y editar entradas de su diario de vuelo**, además de poder visualizarlas todas juntas en una tabla y en detalle individualmente.
- En el **detalle de la entrada de vuelo** aparecerán todos los campos introducidos en el formulario de creación.
- Un usuario registrado podrá **crear y editar rutas**, las cuales tendrán una vista de detalle y podrán verse en las diferentes listas (rutas públicas, rutas creadas y rutas favoritas).
- El **detalle de las rutas** debe contener un mapa con el camino seguido, una descripción, la opción de marcarla como favorita y la opción de descargar el archivo GPX asociado. Además contará con una serie de imágenes, de las cuales se podrá ver su vista en detalle. Por último, se permitirá a los usuarios registrados crear comentarios sobre una ruta pública.
- Un usuario registrado contará con su propio **perfil de usuario**, que contendrá su información personal, además de pestañas con las rutas publicadas, estadísticas de sus vuelos y listas de seguidores y seguidos. Se le debe permitir modificar su información personal.

Todos estos requisitos se han visto reflejados en la pila de producto, la cual se explica en detalle en la Sección 4.1.4.

4.1.3 Requisitos no funcionales

En cuanto a los requisitos no funcionales, se ha creído necesario que la aplicación sea:

- **Segura.** Se considera importante que la aplicación cuente con un sistema de autenticación para que los usuarios no accedan a datos que no les corresponde.
- **Intuitiva y fácil de usar.** Se pretende que los pilotos puedan crear rutas y modificar su diario de vuelo con operaciones sencillas.

4.1.4 Pila de Producto

Una vez recogidos los requisitos descritos en la Sección 4.1.2, se ha entrado en detalle en cada uno de ellos para reflejarlos en una serie de historias de usuario, las cuales se exponen a continuación en orden de desarrollo.

1. Un usuario anónimo accede al formulario de registro y se crea una cuenta. En este formulario se recoge su información personal (Nombre, apellidos, fecha de nacimiento, correo electrónico, ciudad y país) y la contraseña que tendrá que utilizar para iniciar sesión, siendo todos los campos obligatorios.
2. Un usuario anónimo accede al formulario de autenticación para conectarse como usuario registrado, en él deberá introducir su nombre de usuario y su contraseña.
3. Un usuario registrado hace clic en el botón de cerrar sesión y pasa a ser usuario anónimo de nuevo.
4. Un administrador crea una nueva aeronave desde el formulario de creación de aeronaves introduciendo la marca y el modelo.
5. Un administrador accede a una lista con todas las aeronaves creadas.
6. Un administrador elimina una aeronave de las que se encuentran en la lista.
7. Un administrador accede al formulario de edición de una aeronave, el cual contiene los mismos campos que el de creación.
8. Un usuario anónimo que haya olvidado su contraseña puede acceder a un formulario en el que se le pedirá su correo electrónico. Si el correo introducido se encuentra en la base de datos, se le enviará al mismo un enlace personalizado en el que podrá restablecer su contraseña de forma segura.
9. Un administrador crea un nuevo aeródromo desde el formulario de creación de aeródromos introduciendo su nombre, ciudad, país, código IATA, código OACI, elevación y coordenadas. Estas últimas se podrán introducir de forma manual o haciendo clic sobre un mapa.
10. Un administrador accede a la lista de aeródromos creados.
11. Un administrador elimina un aeródromo de los que se encuentran en la lista.
12. Un administrador accede al formulario de edición de un aeródromo, el cual contiene los mismos campos que el formulario de creación.
13. Un usuario registrado o anónimo comprueba la información personal de un perfil de usuario, la cual muestra el nombre completo del piloto, la ciudad indicada en el registro, el país, la fecha de nacimiento, la fecha de registro en la página y una imagen de perfil.
14. Un usuario registrado modifica la información mostrada en su perfil de usuario desde el formulario de edición accesible a través de una pestaña en el propio perfil.

15. Un usuario registrado modifica la imagen de su perfil de usuario, escogiéndola de su ordenador.

16. Usuario registrado crea una entrada del diario de vuelo a través del formulario de creación. Este formulario debe contener los siguientes campos:

    - Aeropuerto de salida
    - Aeropuerto de llegada
    - Día y hora de salida
    - Día y hora de llegada
    - Número de despegues y aterrizajes de día
    - Número de despegues y aterrizajes de noche
    - Tiempo como piloto principal
    - Tiempo como copiloto
    - Tiempo como instructor
    - Tiempo de vuelo dual
    - Tiempo de vuelo de noche
    - Tiempo de vuelo bajo las reglas de vuelo instrumental
    - Tiempo de vuelo con un solo motor
    - Tiempo de vuelo con más de un motor
    - Tiempo de vuelo multipiloto
    - Ruta seguida
    - Aeronave utilizada
    - Observaciones

17. Un usuario registrado visualiza la tabla de entradas del diario de vuelo. Esta tabla contendrá la información básica de cada entrada y opciones para verla en detalle, editarla y eliminarla.

18. Un usuario registrado visualiza el detalle de una entrada del diario de vuelo. Esta pantalla contendrá todos los campos utilizados en la creación del vuelo, además de un mapa mostrando la ruta seguida. En caso de que no se haya seguido ninguna ruta, el mapa mostrará una linea recta entre los aeropuertos si el aeropuerto de salida y el de llegada son distintos, y en caso de que sea el mismo, en el mapa se dibujará un círculo alrededor de éste.

19. Un usuario registrado edita una entrada del diario de vuelo con un formulario similar al de creación.

20. Un usuario registrado elimina una entrada del diario de vuelo desde la tabla de entradas.

21. Un usuario registrado descarga un archivo que contiene el registro de vuelo completo con todos los campos. Dicho archivo tendrá formato CSV y se descargará directamente al pulsar un botón que se encontrará en la misma pantalla que la tabla de entradas.

22. Un usuario registrado o anónimo podrá visualizar estadísticas de los vuelos de cualquier usuario desde una pestaña en el perfil de éste.

23. Un usuario registrado puede seguir a otro usuario pulsando un botón en el perfil del último.

24. Un usuario registrado puede dejar de seguir a otro usuario pulsando un botón en el perfil del último.

25. Un usuario registrado o anónimo puede comprobar la lista de usuarios a los que sigue un piloto desde una pestaña de su perfil. En esta lista se mostrará la imagen de perfil de los pilotos junto a su nombre y su ciudad.

26. Un usuario registrado o anónimo puede comprobar la lista de seguidores de un piloto desde una pestaña de su perfil. En esta lista se mostrará la imagen de perfil de los pilotos junto a su nombre y su ciudad.

27. Un usuario registrado crea una nueva ruta a través del formulario de creación de rutas. En este formulario serán presentarán lo siguientes campos:

    - Nombre de la ruta
    - Aeropuerto de salida
    - Aeropuerto de llegada
    - Descripción de la ruta
    - Importación del archivo GPX de la ruta

    Además de estos campos, se podrán añadir imágenes, las cuales dispondrán de una vista previa en el mismo formulario, y un interruptor para decidir si la ruta es pública o no.

28. Un usuario registrado sube una imagen correspondiente a una ruta a través de un formulario contenido en una ventana modal accesible desde los formularios de creación y edición de las rutas. Para subir la imagen serán imprescindibles un nombre, una descripción, el archivo de la imagen y las coordenadas correspondientes, las cuales se podrán establecer manualmente o buscando el lugar exacto en un mapa.

29. Un usuario registrado elimina una imagen correspondiente a una ruta, ya sea durante la creación o la edición de ésta.

30. Un usuario registrado modifica una ruta de su propiedad a través de un formulario similar al formulario de creación de ruta.

31. Un usuario registrado, anónimo o administrador visualiza una lista de rutas públicas ordenadas por fecha, siendo la última ruta publicada la primera de la lista. En cada ruta se mostrará una imagen de las asociadas a la ruta, el nombre de la ruta, la descripción, el nombre del creador y la fecha de creación de la ruta. En el caso del administrador, en las rutas de la lista se indicará el número de comentarios que contiene para facilitar las labores de moderación.

32. Un usuario registrado visualiza su lista de rutas creadas ordenadas por fecha, siendo la última ruta publicada la primera de la lista, a través de una pestaña en la página de rutas. En cada ruta se mostrará una imagen de las asociadas a la ruta, el nombre de la ruta, la descripción, el nombre del creador y la fecha de creación de la ruta.

33. Un usuario registrado visualiza su lista de rutas favoritas ordenadas por fecha, siendo la última ruta publicada la primera de la lista, a través de una pestaña en la página de rutas. En cada ruta se mostrará una imagen de las asociadas a la ruta, el nombre de la ruta, la descripción, el nombre del creador y la fecha de creación de la ruta.

34. Un usuario registrado o anónimo visualiza la lista de rutas publicadas por un piloto a través de una pestaña de su perfil. Estarán ordenadas por fecha, siendo la última ruta publicada la primera de la lista y mostrarán cuatro imágenes de las asociadas a la ruta, el nombre y la fecha de creación de la ruta. En caso de ser un usuario registrado y estar visualizando su propia lista de rutas, se mostrarán todas las rutas creadas.

35. Un usuario registrado, anónimo o administrador accede a la vista de detalle de una ruta. En esta vista se debe mostrar el nombre de la ruta con su creador y la fecha de creación, un mapa mostrando el camino concreto que sigue la ruta indicado por el archivo GPX importado, una vista previa de las imágenes asociadas a la ruta, así como iconos en el mapa indicando su ubicación, la descripción de la ruta y una sección de comentarios.

36. Un usuario registrado elimina una ruta de su propiedad y, como consecuencia, todas las imágenes asociadas a la misma.

37. Un usuario registrado, a través de la vista de detalle de un vuelo accede a la vista de detalle de la ruta que está asociado a él.

38. Un usuario registrado añade una ruta pública o propia a su lista de rutas favoritas.

39. Un usuario registrado elimina una ruta pública de su lista de rutas favoritas.

40. Un usuario registrado o anónimo descarga el archivo GPX asociado a una ruta.

41. Un usuario registrado accede a los vuelos realizados sobre una ruta concreta.

42. Un usuario registrado o anónimo visualiza en detalle las imágenes asociadas a una ruta. Este detalle se mostrará en una ventana modal que contiene el nombre de la imagen, su descripción y la imagen en cuestión.

43. Un usuario registrado comenta una ruta pública. Este comentario se mostrará en una lista en la sección de comentarios de una imagen y contendrá la imagen de perfil del usuario, su nombre y la fecha de creación del comentario, además del comentario en sí.

44. Un usuario registrado o administrador elimina un comentario. En el caso de ser un comentario de una ruta propia, se podrán borrar los comentarios de otros usuarios, si es una ruta ajena solo se podrán eliminar los comentarios hechos por el usuario autenticado. Un administrador podrá borrar cualquier comentario.

45. Un usuario registrado o anónimo visualiza la página principal de la aplicación. Esta página contendrá una lista con las tres últimas rutas publicadas, de las cuales se mostrará el nombre, tres imágenes, el creador y la fecha de creación, y un mapa con los caminos que siguen esas tres rutas. Además, para los usuarios registrados se mostrarán los tres últimos vuelos realizados, conteniendo cada uno los aeropuertos de salida y llegada, la fecha del vuelo, el avión utilizado y el tiempo total.

46. Un administrados visualiza la página principal de administración de la aplicación, la cual contiene tres enlaces, uno a la lista de aeronaves, otro a la lista de aeródromos y un último enlace a la lista de rutas públicas.

4.2 Arquitectura del sistema

La aplicación se ha construido siguiendo un patrón de arquitectura MVC (Modelo, Vista, Controlador) como se muestra en la Figura 4.1. De esta forma las capas del Servidor y del Cliente Web serán totalmente independientes, lo que facilita su evolución por separado y su flexibilidad a la hora de reutilizarlas. Se comunicarán entre ellas a través de servicios REST.

En la parte del **servidor** se encuentra la lógica de la aplicación. El módulo de acceso a datos se encarga de la comunicación con la base de datos para garantizar la persistencia de la información, mientras que la lógica de negocio provee servicios que contienen las operaciones necesarias para tratar los datos. A estos servicios accederá el Cliente Web a través de una API

REST, la cual expone una serie de *endpoints* a través de los cuales se atenderán las peticiones procedentes del cliente.

Por otro lado, el **Cliente Web** se encargará de presentar al usuario una interfaz gráfica con la que interactuar para realizar todas las operaciones permitidas en la aplicación.

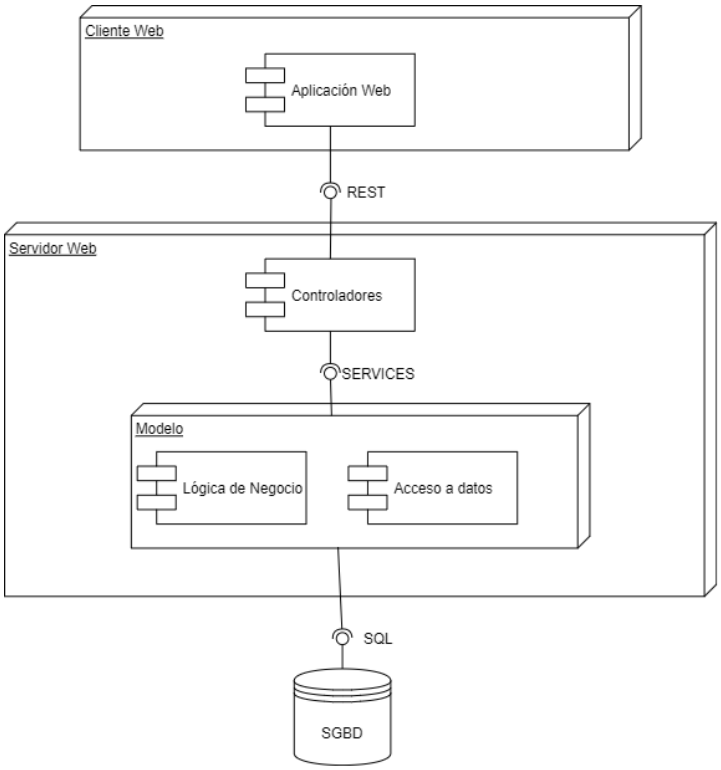


Figura 4.1: Arquitectura MVC de la aplicación

4.3 Interfaz de usuario

En esta sección se realizará una descripción de alto nivel del interfaz de usuario de la aplicación, enumerando las pantallas, describiendo su estructura general e indicando cómo se realiza la navegación entre ellas.

4.3.1 Navegación entre componentes

Se expondrán ahora los distintos componentes que conforman la parte del cliente web, dejando a un lado aquellos que no forman parte de la estructura de pantallas, como los que servirán de apoyo al manejo del estado de la aplicación. Se distinguirán tres grupos distintos de componentes: los utilizados para la página de administración, los que forman parte de la interfaz de los pilotos y los comunes a todos ellos.

Para las **pantallas de administración** se han utilizado los siguientes componentes:

- **HomeAdmin.** La página principal para los administradores. Aquí se mostrarán tres enlaces, uno a la lista de aeronaves, otro a la lista de aeródromos y otro a la lista de rutas.
- **AircraftList.** Lista con todas aeronaves creadas. Cada una tendrá su propio botón de eliminar y otro para editar. Además habrá un botón para crear una nueva aeronave. Pulsando este último botón o el de editar, el componente se cambiará por "AircraftForm".
- **AircraftForm.** Formulario para la creación o edición de aeronaves.
- **AerodromeList.** Lista con los aeródromos creados hasta el momento. Al igual que en la lista de aeronaves, existirá una opción para crear un nuevo aeródromo y cada uno contará con sus botones de editar y eliminar. En caso de pulsar los botones de crear nuevo o editar, el componente se cambiará por "AerodromeForm".
- **AerodromeForm.** Formulario para crear o editar aeródromos. Incluye el componente "MapAerodrome"
- **MapAerodrome.** Componente mostrado dentro de "AerodromeForm" para visualizar un mapa en el cual se pueda indicar la localización del aeródromo, ya sea haciendo clic sobre él o modificando las coordenadas del formulario.

A continuación se muestran los componentes utilizados para las **pantallas de usuarios registrados**:

- **HomePilot.** La página principal para los usuarios registrados. En ella se mostrarán una lista con las tres últimas rutas subidas, desde las cuales se podrá acceder al "RouteDetail" de cada una de ellas, el componente "MapHome", que muestra los caminos seguidos por las rutas de la lista y una lista con los tres últimos vuelos realizados, desde los cuales se podrá acceder al "FlightDetail" de cada uno.
- **ProfileEdit.** Componente para editar la información personal y la contraseña del piloto. Se accede a través de una pestaña del componente "Profile".
- **FlightList.** Muestra una tabla con las entradas del diario de vuelo del piloto registrado. En cada entrada de la tabla existen tres iconos de acción: uno para acceder a la vista de detalle del vuelo en el componente "FlightDetail, otro para editar la entrada con "FlightForm" y un último para eliminar la entrada. Además en el componente se muestran otros dos botones: uno para crear una nueva entrada con "FlightForm" y otro para descargar la lista en formato CSV. Existirá también un buscador para filtrar las entradas de la tabla.

- **FlightForm.** Formulario para la creación y edición de entradas del diario de vuelo.
- **FlightDetail.** Componente que muestra los detalles de una entrada de vuelo, indicando todos los campos utilizados para su creación, además del componente "MapFlight", que muestra la ruta seguida por el vuelo.
- **MapFlight.** Componente incluido dentro de "FlightDetail" que muestra un mapa con el camino seguido por el vuelo. En caso de que el vuelo siga una ruta, dibuja el camino exacto de dicha ruta, si no sigue ninguna ruta se contemplan dos posibilidades: si el aeropuerto de salida y el de llegada son el mismo, dibuja un círculo alrededor de este, y si son distintos, dibuja una linea recta entre los dos.
- **RouteListCreated.** Incluido en el componente "RouteList". Muestra una lista con las rutas creadas por el usuario identificado, desde las cuales se podrá acceder al "RouteDetail" de cada una.
- **RouteListFav.** Incluido en el componente "RouteList". Muestra una lista con las rutas que el usuario identificado ha marcado como favoritas, desde las cuales se podrá acceder al "RouteDetail" de cada una.
- **RouteForm.** Formulario para la creación y edición de rutas. Incluye un botón para acceder al componente "NewImage".
- **NewImage.** Formulario para la inclusión de una imagen en la creación o edición de rutas. Incluye el componente "MapImage".
- **MapImage.** Componente incluido dentro de "NewImage". Muestra un mapa en el que se indica la ubicación exacta de la imagen que se ha subido. Esta ubicación se puede modificar haciendo clic en el mapa o cambiando las coordenadas en el componente padre.

Por último, se indican los **componentes comunes** a todos los usuarios:

- **NavBar.** Componente que muestra en todo momento el menú de la aplicación. En caso de ser administrador, se mostrarán enlaces a "HomeAdmin", "AerodromeList", "AircraftList" y "RouteList", además de dar la posibilidad de cerrar sesión. Si no hay ningún usuario identificado, se muestran enlaces a "HomeAnonymous", "RouteList", "SignIn" y "Register". Por último si el usuario es un piloto identificado, el componente mostrará enlaces a "HomePilot", "FlightList", "RouteList", "Profile", además de dar la posibilidad de cerrar sesión.
- **Page404.** Este componente mostrará un mensaje de error en caso de que el enlace al que se intenta acceder no esté registrado en la aplicación.

- **HomeAnonymous.** Página principal para los usuarios no identificados. Muestra un mensaje de bienvenida en el que se da la posibilidad de acceder al componente "SignIn" para identificarse como piloto o al componente "Register" para crear una cuenta. Además, se incluye una lista con las tres últimas rutas publicadas, a través de las cuales se podrá acceder al "RouteDetail" de cada una.

- **MapHome.** Mapa que se incluye en los componentes "HomePilot" y "HomeAnonymous" y muestra los caminos seguidos por las rutas de la lista.

- **Register.** Componente que incluye un formulario para la creación de una cuenta de piloto.

- **SignIn.** Componente para la identificación de un usuario anónimo como piloto. Incluye un botón que redirige al componente "Register" y un enlace al componente "ForgotPass" en caso de que el usuario haya olvidado su contraseña.

- **ForgotPass.** En este componente da la opción al usuario de introducir su correo electrónico para enviarle un enlace y restablecer su contraseña. Una vez enviado se le redirige al componente "SignIn".

- **ResetPass.** Es el único componente al que no se puede acceder directamente desde la aplicación, solamente se puede acceder a través de un enlace en el correo electrónico de un usuario registrado o administrador. Mostrará un formulario para restablecer la contraseña del usuario. Una vez restablecida, se le redirigirá al componente "SignIn".

- **Profile.** Componente que muestra el perfil de un piloto registrado. Contendrá una serie de pestañas y mostrará distintos componentes dependiendo de la pestaña marcada. Si se marca la pestaña de "info", mostrará el componente "ProfileInfo", si se marca la pestaña "routes", mostrará el componente "ProfileRoutes" y en caso de marcar la pestaña "statistics", se mostrará el componente "ProfileStatistics". También incluye dos enlaces que cambiaran el contenido al componente "ProfileFollowers", mostrando las listas de seguidores y seguidos. En caso de que se trate del perfil del usuario identificado, también se mostrará una pestaña que, al marcarla, mostrará el componente "ProfileEdit".

- **ProfileInfo.** Contenido dentro del componente "Profile". Muestra la información personal del piloto. En caso de tratarse del perfil del usuario identificado, se mostrará un botón para modificar la imagen de perfil.

- **ProfileRoutes.** Muestra una lista con las rutas publicadas por el piloto. En caso de ser el perfil del usuario identificado, se mostrarán todas las rutas creadas. Se podrá hacer clic en cada ruta para acceder al componente "RouteDetail".

- **ProfileStatistics.** Muestra un pequeño conjunto de estadísticas de los vuelos y rutas del piloto.

- **ProfileFollowers.** Contiene una lista de los pilotos seguidos o a los que sigue (dependiendo del enlace desde el que se haya accedido) el usuario del perfil. Se puede acceder haciendo clic al perfil de cada piloto de la lista.

- **RouteDetail.** Muestra el detalle de una ruta. Contendrá un enlace al perfil del piloto que ha creado la ruta. Si el usuario identificado es el creador de la ruta se mostrarán cinco botones: uno para eliminar la ruta, otro para editarla con el componente "RouteForm", otro para marcarla como favorita, otro para acceder al componente "FlightList" con la búsqueda inicializada al nombre de la ruta, lo que nos mostrará todos los vuelos realizados sobre dicha ruta, y un último botón para descargarla en un formato adecuado para ser compartido. En caso de que el usuario identificado no sea el creador se omitirán los botones de eliminar y editar y en el caso de que se trate de un usuario anónimo o un administrador solamente se mostrará el de descargar la ruta. Además, se mostrará el componente "MapRoute" con el camino seguido por la ruta y las ubicaciones de las imágenes, se podrá acceder al detalle de cada imagen, y se podrán escribir comentarios.

- **MapRoute.** Componente que muestra un mapa con el camino seguido por una ruta. Además, se indicarán los aeródromos de salida y llegada y la ubicación exacta de cada imagen, pudiendo hacer clic en los iconos para ver el nombre de la imagen y una vista previa.

4.3.2 Maquetas de los componentes

En esta sección se incluirán algunos de los *mockups* de los componentes que se han considerado más importantes para entender la estructura básica de la aplicación y de las pantallas que contendrán dichos componentes. Este tipo de maquetas se crean antes de comenzar el desarrollo de la aplicación para tener una idea inicial y no tener que estructurarlo todo sobre la marcha. Se puede ver todas las maquetas creadas en el Anexo B.

En la Figura 4.2 se observa la pantalla principal que se le mostrará a los usuarios registrados y anónimos. Aunque ha sufrido pequeños cambios que se han considerado necesarios a lo largo del desarrollo (se decidió que para los usuarios identificados mostrase también los tres últimos vuelos realizados), se puede observar la estructura básica, la cual se repite en el resto de pantallas. En la parte superior se observa el menú de navegación, en el que se encuentran los enlaces a las listas de rutas y vuelos en el caso de los usuarios registrados. A la izquierda del menú hay un enlace a la página principal, el cual se ha sustituido por el logo de la aplicación. En la parte derecha hay un enlace al perfil de usuario, en caso de estar identificado y la

opción de cerrar sesión. Para los usuarios anónimos ahí habrá un enlace a la página de inicio de sesión y otro a la de registro. Debajo del menú se encuentra el contenido de esta página, que en este caso está formado por el componente "HomePilot" mencionado en la Sección 4.3.1. En componente está formado por una lista de rutas a la izquierda y un mapa con los caminos que siguen las rutas de la lista a la derecha. Haciendo clic en una ruta de la lista el usuario será redirigido a la vista de detalle de la misma, mostrada en la Figura 4.5.

Figura 4.2: Maqueta de la pantalla principal de los pilotos

Figura 4.3: Maqueta de la pantalla de creación de ruta

La Figura 4.3 muestra la pantalla de creación de una nueva ruta. En ella se deberán incluir los distintos atributos necesarios para la creación de la ruta como el nombre, una descripción, los aeropuertos de salida y llegada, que se seleccionarán de una lista de los aeropuertos existentes. Habrá un interruptor para decidir si la ruta es pública o no. Aunque en la maqueta inicial existía un mapa para mostrar el recorrido de la ruta, se ha cambiado para importar un archivo GPX, el cual contiene esta información. Además, se podrán subir seis imágenes,

las cuales dispondrán de una vista previa en el formulario, como se puede ver en la Figura mencionada. Si se pulsa el botón de "Añadir imágenes" se abrirá una ventana modal, como muestra la Figura 4.4, en la que aparecerán los campos necesarios para subir una imagen (nombre, descripción, coordenadas y archivo de imagen).

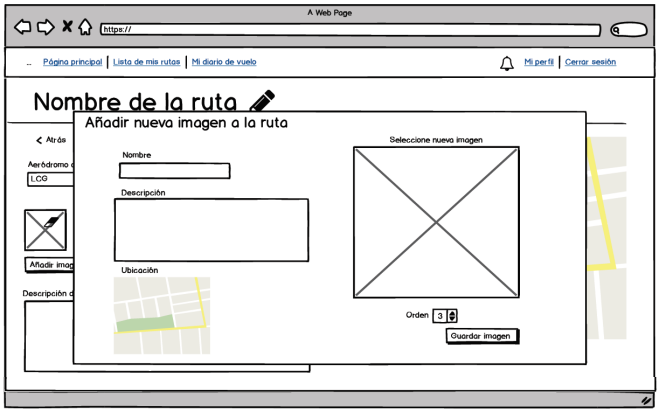


Figura 4.4: Maqueta de la pantalla de nueva imagen

Una vez creada la ruta, ésta se podrá revisar en la pantalla de detalle de ruta, de la cual muestra su maqueta la Figura 4.5. En ella se mostrarán su nombre y autor en la parte del título. En el contenido se encontrarán los botones de descargar ruta, marcar como favorita y visualizar los vuelos realizados sobre esa ruta en la parte de arriba (en caso de ser una ruta propia aparecerán también los de editar y eliminar). También se mostrará un mapa con el recorrido de la ruta y las ubicaciones de las imágenes, una descripción y los aeropuertos de salida y llegada. Por último, una sección de comentarios en la que cualquier usuario registrado podrá dejar su opinión acerca de la ruta.

Figura 4.5: Maqueta de la pantalla de detalle de ruta

4.4 Modelo conceptual de datos

En esta sección se mostrarán las distintas entidades que formarán la base de datos de la aplicación y sus atributos.

User representa a los usuarios conectados a la aplicación, ya sean pilotos o administradores. Para estos últimos, esta entidad contiene toda la información necesaria. Tiene los siguientes atributos:

- *idUser:* Identificador del usuario.
- *login:* Login del usuario.
- *password:* Contraseña del usuario.
- *authority:* Autoridad del usuario, puede ser PILOT o ADMIN.
- *email:* Correo electrónico del usuario.

UserAuthority es un enumerado para representar los permisos del usuario (PILOT para los usuarios registrados y ADMIN para los administradores).

Pilot representa a los usuarios registrados en la aplicación. Se compone de los atributos siguientes:

- *name:* Nombre del piloto.
- *surname1:* Primer apellido del piloto.
- *surname2:* Segundo apellido del piloto.
- *city:* Ciudad de residencia del piloto.
- *country:* País de residencia del piloto.
- *birthDate:* Fecha de nacimiento del piloto.
- *regisDate:* Fecha en la que el piloto se registró en la aplicación.
- *followers:* Lista de los pilotos que le siguen.
- *following:* Lista de pilotos a los que sigue.
- *createdRoutes:* Lista de rutas creadas por el piloto.
- *favRoutes:* Lista de rutas favoritas del piloto.

Aircraft representa a las aeronaves incluidas por el administrador. Contará con los siguientes atributos:

- *idAircraft:* Identificador de la aeronave.

- *manufacturer:* El fabricante de la aeronave.
- *model:* El modelo de la aeronave.

Aerodrome se refere a los aeródromos creados por el administrador. Está formado por los siguientes atributos:

- *idAerodrome:* Identificador del aeródromo.
- *name:* Nombre del aeródromo.
- *city:* Ciudad donde se encuentra el aeródromo.
- *country:* País donde se encuentra la ciudad del aeródromo.
- *codIATA:* Código identificador de IATA[1].
- *codOACI:* Código identificador de OACI [2].
- *elevation:* Altura a la que se encuentra el aeródromo en pies [3].
- *position:* Coordenadas de la ubicación exacta del aeropuerto.

Route representa una ruta creada por un piloto. Se compondrá de las siguientes propiedades:

- *idRoute:* Identificador de la ruta.
- *name:* Nombre de la ruta.
- *isPublic:* Indica si una ruta es pública o no.
- *description:* Breve descripción de la ruta.
- *creationDay:* Fecha en que la ruta fue creada.
- *path:* Serie de coordenadas que representan el recorrido de la ruta.
- *takeoffAerodrome:* Aeropuerto de salida de la ruta.
- *landingAerodrome:* Aeropuerto de llegada de la ruta.
- *pilot:* Usuario creador de la ruta.
- *flights:* Lista de vuelos que siguen esta ruta.
- *comments:* Lista de comentarios sobre la ruta.
- *images:* Lista de imágenes asociadas a la ruta.

[1]Código de tres letras asignado por la Asociación Internacional de Transporte Aéreo.
[2]Código de cuatro caracteres alfanuméricos asignado por la Organización de Aviación Civil Internacional.
[3]El pie es la unidad de longitud utilizada en aviación. Un pie corresponde a 0,3048 metros.

Comments son los comentarios que los pilotos escribirán para cada ruta. Contará con los siguientes atributos:

- *idCommetn:* Identificador del comentario.
- *description:* Texto del comentario.
- *date:* Fecha de creación del comentario.
- *route:* Ruta que se comenta.
- *pilot:* Piloto que escribe el comentario.

Images representa a las imágenes asociadas a las rutas, las cuales se compondrán de los siguientes atributos.

- *idImage:* Identificador de la imagen.
- *name:* Nombre de la imagen.
- *description:* Breve descripción de la imagen.
- *position:* Coordenadas que indican la ubicación de la imagen.
- *route:* Ruta a la que pertenece la imagen.

Por último, ***Flight*** representará cada vuelo realizado por un piloto. Contará con las siguientes propiedades:

- *idFlight:* Identificador del vuelo.
- *departureDate:* Día de salida del vuelo.
- *departureTime:* Hora de salida del vuelo.
- *arrivalDate:* Día de llegada del vuelo.
- *arrivalTime:* Hora de llegada del vuelo.
- *totalTime:* Tiempo total de vuelo.
- *seTime:* Tiempo de vuelo con un solo motor.
- *meTime:* Tiempo de vuelo con más de un motor.
- *mpTime:* Tiempo de vuelo multipiloto.
- *takeoffsDay:* Número de despegues de día.
- *takeoffsNight:* Número de despegues de noche.
- *landingsDay:* Número de aterrizajes de día.
- *landingsNight:* Número de aterrizajes de noche.

- *nightTime:* Tiempo de vuelo nocturno.
- *ifrTime:* Tiempo de vuelo bajo las reglas de vuelo instrumental.
- *picTime:* Tiempo de vuelo como piloto al cargo.
- *coopilotTime:* Tiempo de vuelo como copiloto.
- *dualTime:* Tiempo de vuelo con pilotaje dual.
- *instructorTime:* Tiempo de vuelo como instructor.
- *observations:* Observaciones acerca del vuelo.
- *aircraft:* Aeronave utilizada en el vuelo.
- *aircraftReg:* Matrícula de la aeronave utilizada en el vuelo.
- *takeoffAerodrome:* Aeródromo utilizado para el despegue.
- *landingAerodrome:* Aeródromo utilizado para el aterrizaje.
- *picUser:* Usuario creador del vuelo.
- *route:* Ruta seguida en el vuelo.

En la Figura 4.6 se muestra un diagrama de clases que representa las entidades nombradas en esta sección, sus atributos con sus tipos de datos y las relaciones entre ellas.

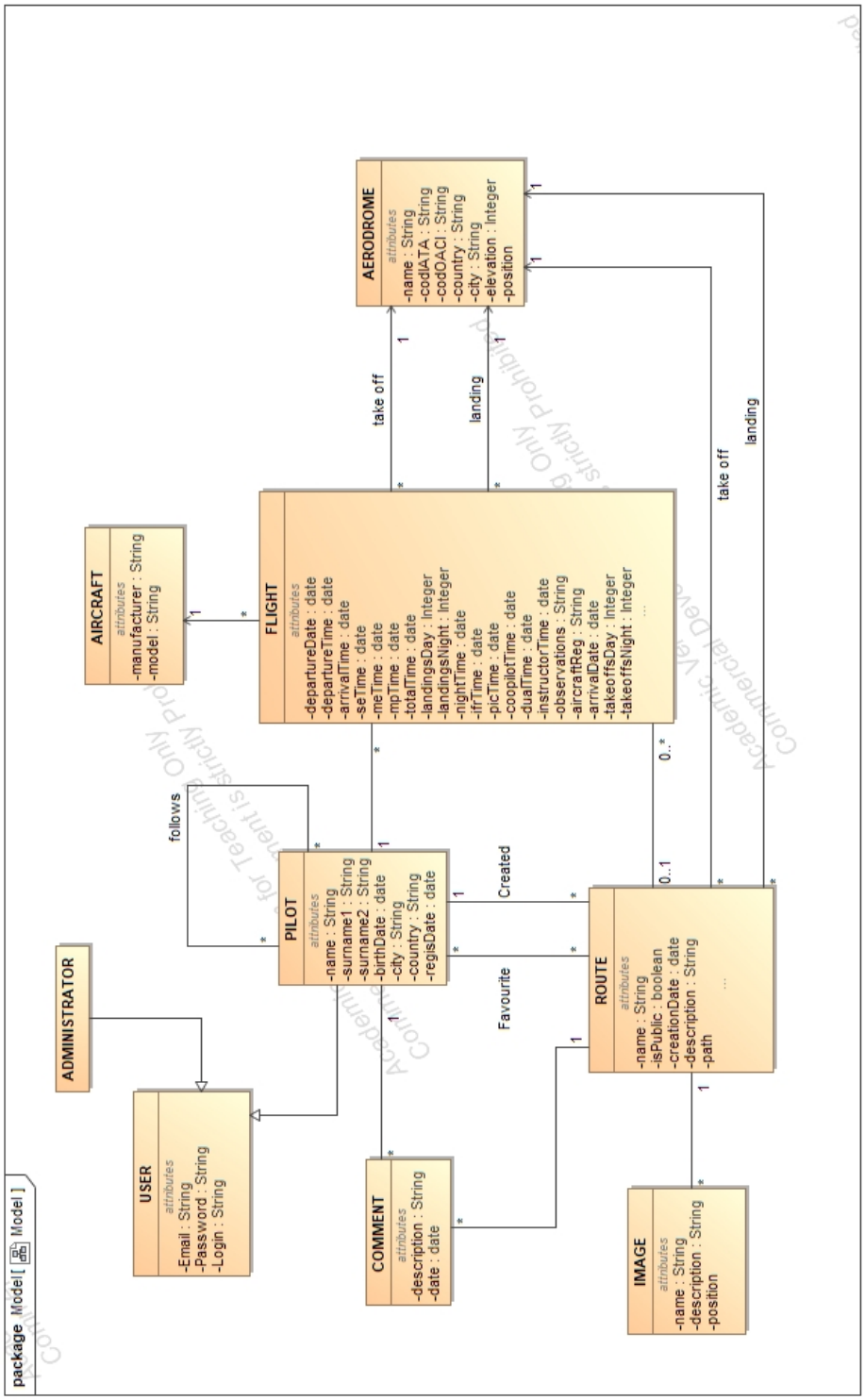


Figura 4.6: Diagrama de clases

Capítulo 5

Diseño

5.1 Arquitectura tecnológica del sistema

Se realizará ahora una descripción de la arquitectura tecnológica utilizada en la aplicación, en la cuál se explicarán las tecnologías que componen cada parte de la Figura 4.1, expuesta en la Sección 4.2. Se hará en un principio una distinción entre las tres capas más diferenciadas de la arquitectura: la parte del almacenamiento de datos, el servidor y el cliente.

Para el **almacenamiento de datos**, se ha utilizado PostgreSQL, un sistema de gestión de bases de datos relacional y orientado a objetos. El motivo principal de la elección de este gestor es que incluye PostGIS, una extensión que convierte el sistema en una base de datos espacial mediante la adición de tipos de datos espaciales que dan soporte a la necesidad de almacenar las ubicaciones de imágenes y aeródromos y el recorrido de las rutas, el cuál viene dado por el archivo GPX asociado a las mismas. Se utilizará este formato porque es compatible con la mayoría de GPS del mercado.

La capa de almacenamiento de datos se comunicará con la parte del **servidor** a través de clases que utilizan el patrón DAO (Data Access Object). A su vez, ésta se comunican con los servicios implementados con Spring e Hibernate, donde se llevarán a cabo las distintas operaciones permitidas por la aplicación.

Esta última capa expondrá un servicio REST, el cual es capaz de comunicar dos sistemas basados en el protocolo HTTP. Se utilizará este servicio REST para asegurar la comunicación con el **cliente**, que se corresponde con una aplicación Web implementada con React, una librería de JavaScript de código abierto para desarrollar interfaces de usuario mediante componentes y Redux, que controla el estado y la persistencia de la aplicación. El cliente Web hará uso de Bootstrap para el diseño de las pantallas y de Leaflet, una librería de JavaScript para visualizar mapas e interactuar con ellos.

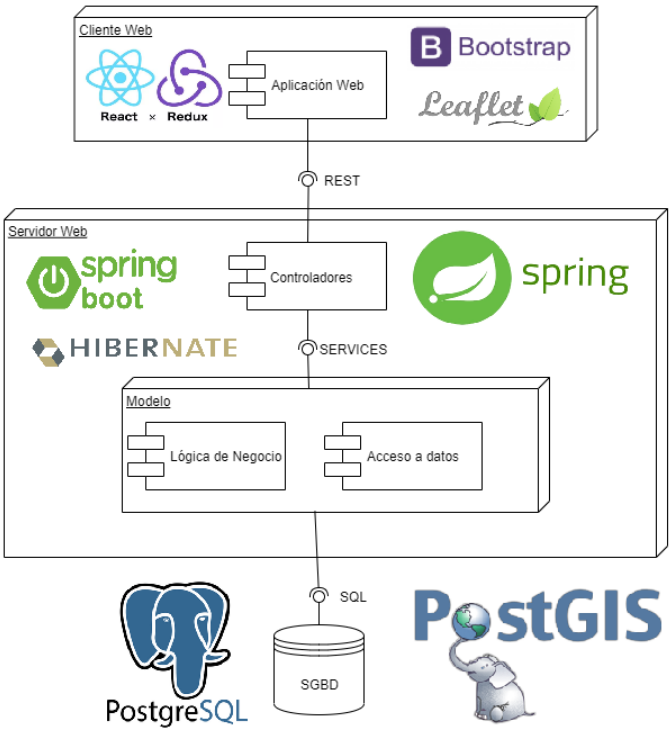


Figura 5.1: Diagrama de componentes con tecnologías.

5.2 Diseño de la aplicación

En esta sección se hará una descripción detallada de las partes de la aplicación descritas en la Sección 5.1. Para ello haremos distinción entre la parte del servidor y la parte del cliente Web.

5.2.1 Servidor

En esta sección se desgranarán las capas que forman parte del *back-end* de la aplicación, excluyendo la capa de persistencia explicada en la Sección 5.1. Se compondrá por tanto por la capa de acceso a datos, la de servicios y los controladores para la comunicación con el cliente.

La **capa de acceso a datos** utiliza el patrón DAO (Data Access Object) para comunicarse con el gestor de almacenamiento. Este patrón propone separar por completo la lógica de negocio de la de acceso a datos proporcionando los métodos necesarios para insertar, actualizar, eliminar y consultar la información almacenada en la base de datos, dejando a la capa de servicios el resto de operaciones necesarias.

Existirá un repositorio por cada entidad, cada uno con un fichero de interfaz y otro de implementación. En cada uno se utilizarán los métodos CRUD proporcionados por *Hibernate*,

además de operaciones de recuperación de información creadas por el autor con el lenguaje HQL (Hibernate Query Language) como el ejemplo de la Figura 5.2.

```
@Override
public Aerodrome findById(Long id) {
    return (Aerodrome) getSession().createQuery("from Aerodrome a where a.id = :id").setParameter("id", id).uniqueResult();
}
```

Figura 5.2: Ejemplo de consulta HQL.

En la capa de **servicios** se encuentra la lógica de negocio de la aplicación. En ella se implementan las operaciones que se especificaron en el análisis de los requisitos. Estos servicios se comunicarán por una parte con la capa de acceso a datos explicada anteriormente para trabajar con los datos necesarios en cada operación, y por otra con los controladores que le piden la información requerida por el cliente Web. En el proyecto, cada entidad cuenta con su propio servicio, los cuales estarán contenidos en la carpeta *es.udc.lbd.asi.restexample.model.service* y su contenido se expone a continuación:

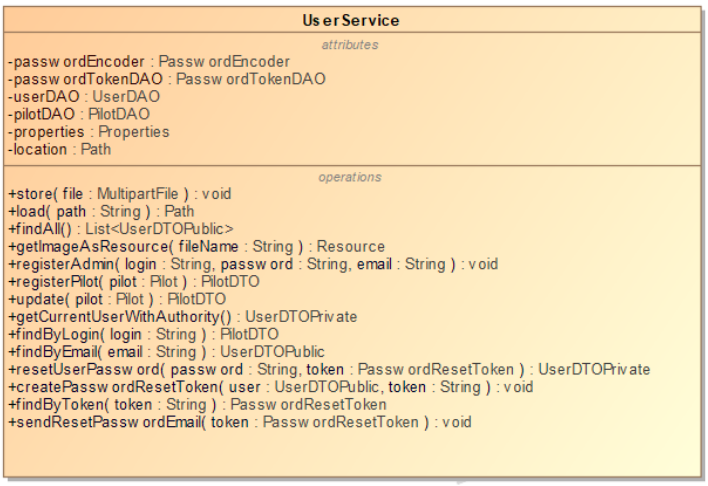


Figura 5.3: UserService.

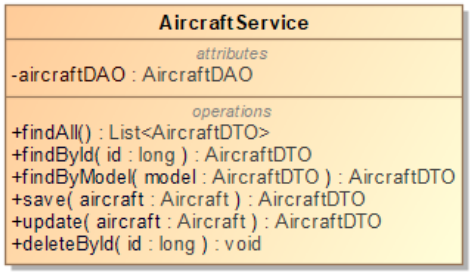


Figura 5.4: AircraftService.

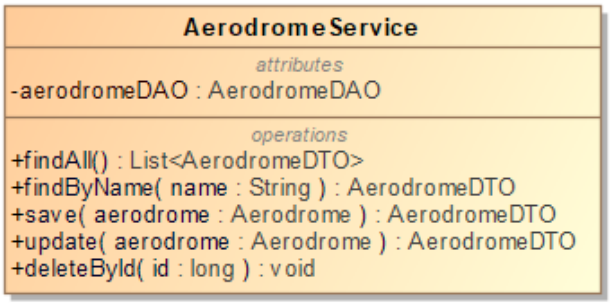


Figura 5.5: AerodromeService.

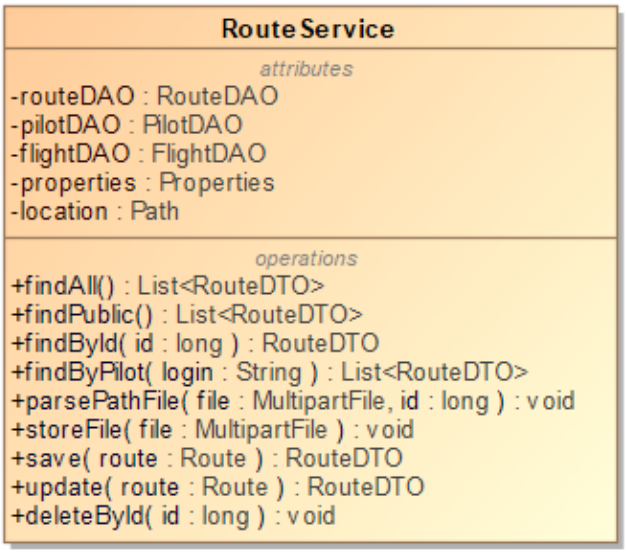


Figura 5.6: RouteService.

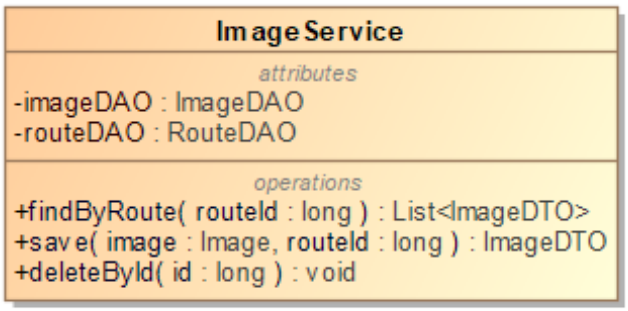


Figura 5.7: ImageService.

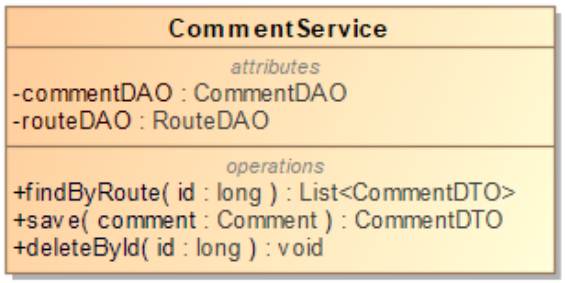


Figura 5.8: CommentService.

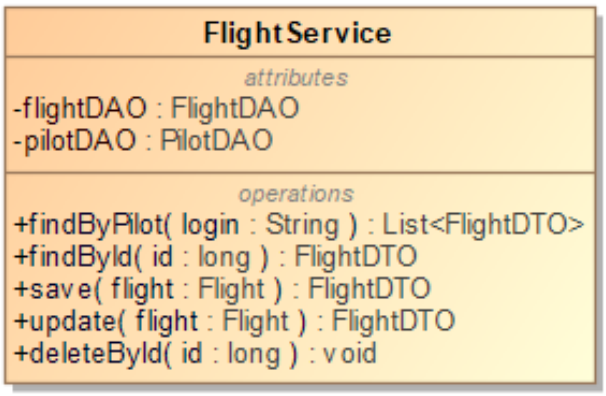


Figura 5.9: FlightService.

Como se ha descrito anteriormente, estos servicios estarán comunicados directamente con los **controladores** que exponen los distintos *endpoints* necesarios para la comunicación con el cliente. Dichos controladores se encuentran en la carpeta *es.udc.lbd.asi.restexample.web* del proyecto y se incluye uno por cada entidad, además de uno a mayores para las cuentas de usuario. A continuación se detallan qué métodos incluirá el controlador de cada entidad, la operación REST necesaria para llevarlo a cabo y las URL para acceder a ellos:

- **Controlador para cuentas.** Contiene los métodos encargados de gestionar el registro de cuentas, así como la autenticación y la restauración de contraseñas.

OPERACIÓN	URL	MÉTODO
POST	/api/authenticate	authenticate
POST	/api/register	registerAccount
POST	/api/forgotpassword	forgotPassword
PUT	/api/resetpassword/token	resetPassword
GET	/api/account	getAccount

Cuadro 5.1: AccountResource

- **Controlador para usuarios.** Contiene los métodos encargados de gestionar los usuarios. Se tratan operaciones de recuperación de uno o varios usuarios, y modificación de imagen de perfil e información personal.

OPERACIÓN	URL	MÉTODO
GET	/api/users	findAll
GET	/api/users/login	findByLogin
GET	/api/users/image/path	getImage
PUT	/api/users/updateavatar/login	updatePilotAvatar
PUT	/api/users/login	updatePilot

Cuadro 5.2: UserResource

- **Controlador para aeronaves.** Contiene las operaciones necesarias para la gestión de aeronaves. Creación, modificación, borrado y recuperación de una o varias.

OPERACIÓN	URL	MÉTODO
GET	/api/aircraft	findAll
GET	/api/aircraft/model	findByModel
POST	/api/aircraft	saveAircraft
PUT	/api/aircraft/id	updateAircraft
DELETE	/api/aircraft/id	deleteAircraft

Cuadro 5.3: AircraftResource

- **Controlador para aeródromos.** Contiene los métodos necesarios para gestionar los aeródromos. Consta de las mismas operaciones que las descritas para las aeronaves.

OPERACIÓN	URL	MÉTODO
GET	/api/aerodromes	findAll
GET	/api/aerodromes/name	findByName
POST	/api/aerodromes	saveAerodrome
PUT	/api/aerodromes/id	updateAerodrome
DELETE	/api/aerodromes/id	deleteAerodrome

Cuadro 5.4: AerodromeResource

- **Controlador para rutas.** Se compone de los métodos necesarios para la gestión de rutas, desde los necesarios para crear, editar y eliminar rutas hasta los de recuperación dependiendo de la información que se le pase al controlador. Además contiene la operación necesaria para descargar el archivo GPX asociado a la ruta indicada por el cliente.

OPERACIÓN	URL	MÉTODO
GET	/api/routes	findPublic
GET	/api/routes/login	findByPilot
GET	/api/routes/detail/id	findById
GET	/api/routes/downloadgpx/id	downloadGPX
POST	/api/routes	saveRoute
PUT	/api/routes/id	updateRoute
PUT	/api/routes/uploadfiles/id	uploadRouteFiles
DELETE	/api/routes/id	deleteRoute

Cuadro 5.5: RouteResource

- **Controlador para las imágenes de ruta.** Contiene los métodos necesarios para la gestión de las imágenes asociadas a una ruta.

OPERACIÓN	URL	MÉTODO
GET	/api/images/routeId	findByRoute
POST	/api/images/routeId	saveImage
DELETE	/api/images/id	deleteImage

Cuadro 5.6: ImageResource

- **Controlador para los comentarios.** Contiene los métodos que se necesitan para gestionar los comentarios de las rutas.

OPERACIÓN	URL	MÉTODO
GET	/api/comments/routeId	findByRoute
POST	/api/comments	saveComment
DELETE	/api/comments/id	deleteComment

Cuadro 5.7: CommentResource

- **Controlador para vuelos.** Contiene los métodos necesarios para gestionar las entradas del diario de vuelo.

OPERACIÓN	URL	MÉTODO
GET	/api/flights/login	findByPilot
GET	/api/flights/details/id	findById
POST	/api/flights	saveFlight
PUT	/api/flights/id	updateFlight
DELETE	/api/flights/id	deleteFlight

Cuadro 5.8: FlightResource

5.2.2 Cliente

Para la parte del **cliente** de la aplicación se ha creado un cliente Web con React, una librería de JavaScript de código abierto que facilita la creación de interfaces de usuario a través de componentes interactivos y reutilizables. Una de las características principales de esta librería es que dipone de un DOM (Document Object Model) virtual, el cual es capaz de reconocer las modificaciones en la página y actualizar solamente las partes que han cambiado, lo que implica que no se tendrá que recargar totalmente la página para mostrar los cambios.

Como se ha indicado, las interfaces estarán creadas a partir de **componentes**, los cuales se pueden crear a partir de funciones simples o a través de clases de ES6[1]. Éstos aceptan propiedades en su definición y pueden contener otros componentes, así como extender a otros componentes base.

En la Figura 5.10 se muestra cómo están distribuidos los componentes del cliente Web según la distinción por usuarios hecha en la Sección 4.3.1.

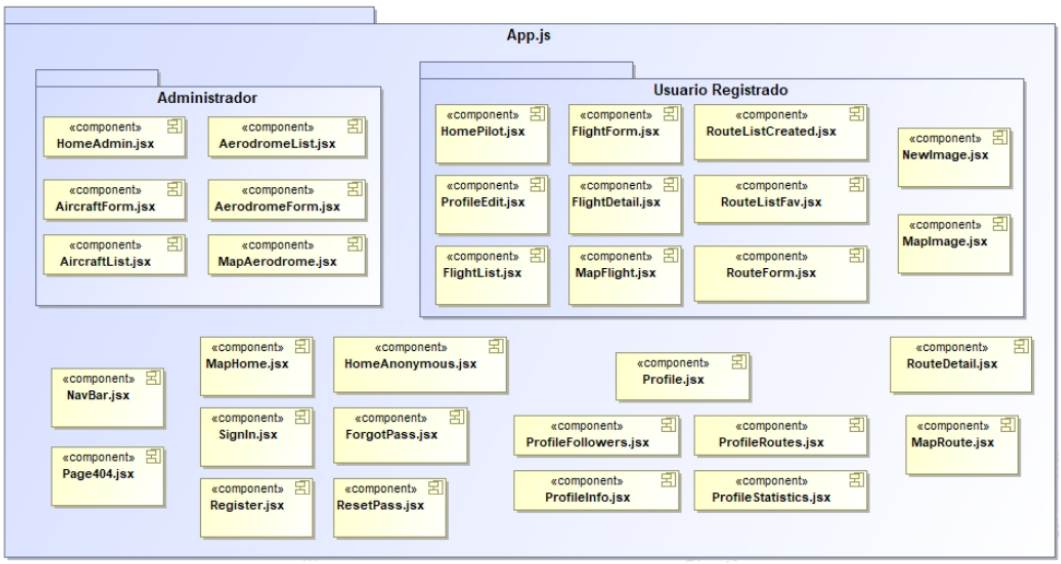


Figura 5.10: Componentes del cliente Web.

Para que el cliente muestre una pantalla u otra, se ha creado el fichero *routes.js*, en el que la librería *react-router-dom* y *Redux* trabajarán al unísono para ofrecer los componentes deseados según la **ruta** indicada en el navegador.

De nuevo distinguiendo los distintos usuarios que utilizarán la aplicación, en los Cuadros 5.9, 5.10 y 5.11 se muestran las rutas disponibles y los componentes que contendrán cada una de ellas, omitiendo los que no se muestran en una ruta específica.

[1]ES6 (ECMAScript v6) es el estándar que sigue JavaScript desde Junio del 2015 creado por ECMA International.

ADMINISTRADOR	
RUTA	**COMPONENTES**
/	HomeAdmin.jsx
/aircraft	AircraftList.jsx
/aircraft/new	AircraftForm.jsx
/aircraft/edit/id	AircraftForm.jsx
/aerodromes	AerodromeList.jsx
/aerodromes/new	AerodromeForm.jsx | MapAerodrome.jsx
/aerodromes/edit/id	AerodromeForm.jsx | MapAerodrome.jsx

Cuadro 5.9: Rutas para un usuario administrador.

USUARIO REGISTRADO	
RUTA	**COMPONENTES**
/	HomePilot.jsx | MapHome.jsx
/profile/:login	Profile.jsx | ProfileInfo.jsx | ProfileStatistics.jsx | ProfileRoutes.jsx | ProfileFollowers.jsx | ProfileEdit.jsx
/myLogbook	FlightList.jsx
/flights/new	FlightForm.jsx
/flights/edit/:id	FlightForm.jsx
/flights/:id	FlightDetail.jsx | MapFlight.jsx
/routes	RouteListPublic.jsx | RouteListCreated.jsx | RouteListFav.jsx
/routes/new	RouteForm.jsx | NewImage.jsx | MapImage.jsx
/routes/edit/:id	RouteForm.jsx | NewImage.jsx | MapImage.jsx

Cuadro 5.10: Rutas para un usuario registrado.

USUARIO ANÓNIMO Y COMUNES	
RUTA	**COMPONENTES**
/	HomeAnonymous.jsx | MapHome.jsx
/profile/:login	Profile.jsx | ProfileInfo.jsx | ProfileStatistics.jsx | ProfileRoutes.jsx | ProfileFollowers.jsx
/routes	RouteListPublic.jsx
/routes/:id	RouteDetail.jsx | MapRoute.jsx
/signIn	SignIn.jsx
/register	Register.jsx
/forgotPassword	ForgotPass.jsx
/ResetPassword/:token	ResetPass.jsx

Cuadro 5.11: Rutas para un usuario anónimo y comunes.

Redux es una herramienta para gestionar el estado de aplicaciones JavaScript. Implementa el patrón *Flux*, el cual almacena el estado global de la aplicación en una única *Store*, que se va modificando según se lo indiquen los componentes de la vista a través de acciones. Redux

se ha utilizado para controlar qué usuario está autenticado en cada momento y para que ese usuario reciba la información que le corresponde proveniente del servidor.

Para la **comunicación con el servidor** se realizarán peticiones HTTP a cada uno de los controladores con métodos REST del servidor. Estas peticiones se harán a través de *Axios*, una API HTTP basada en *XMLHttpRequest*. Estas peticiones tendrán la apariencia de la Figura 5.11.

```
HTTP.get("/aerodromes")
  .then(res => {
    const aerodromes = res.data;
    this.props.initializeAerAction(aerodromes);
    this.setState({ aerodromes: res.data });
  })
  .catch(function(error) {
    notify.show("Error in get aerodromes", "error", 2000);
  });
```

Figura 5.11: Petición HTTP GET para aeródromos.

A continuación se muestran en las Figuras 5.12, 5.13, 5.14 y 5.15 diagramas para ilustrar con qué controlador del servidor se comunica cada uno de los componentes del cliente Web que necesita acceder a operaciones del servidor.

- Para los componentes exclusivos del **administrador**:

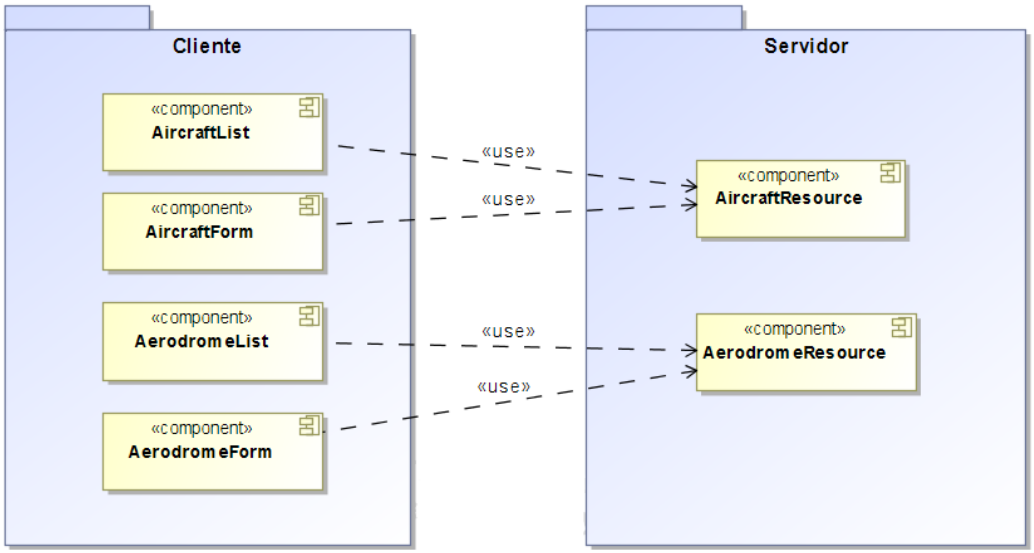


Figura 5.12: Comunicación entre los componentes del administrador y el servidor.

- Para los componentes exclusivos de los **usuarios registrados** se han dividido en dos: La Figura 5.13 muestra los componentes relativos a las rutas y la Figura 5.14 los relativos a los vuelos, la página principal y la edición del perfil.

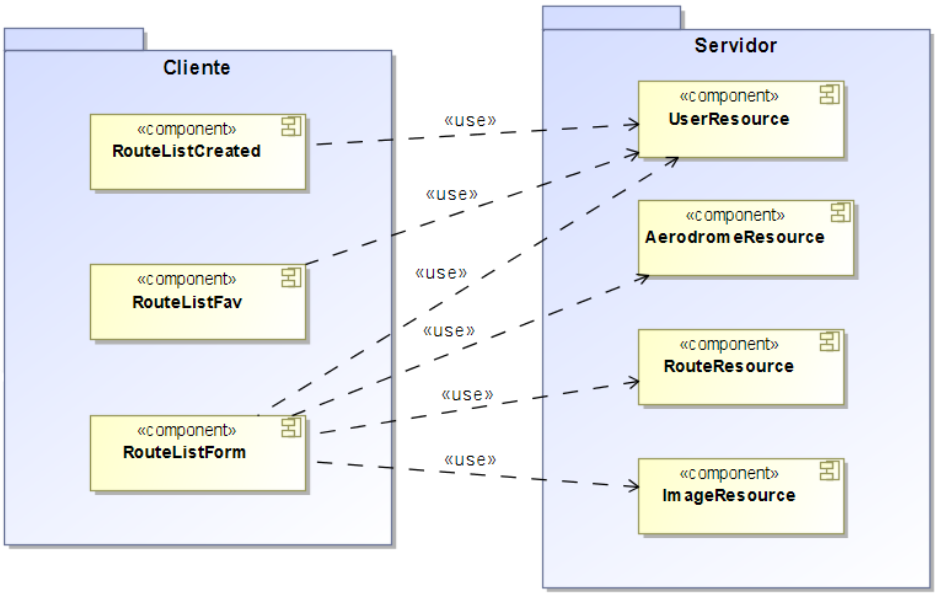


Figura 5.13: Comunicación entre los componentes de pilotos y el servidor.

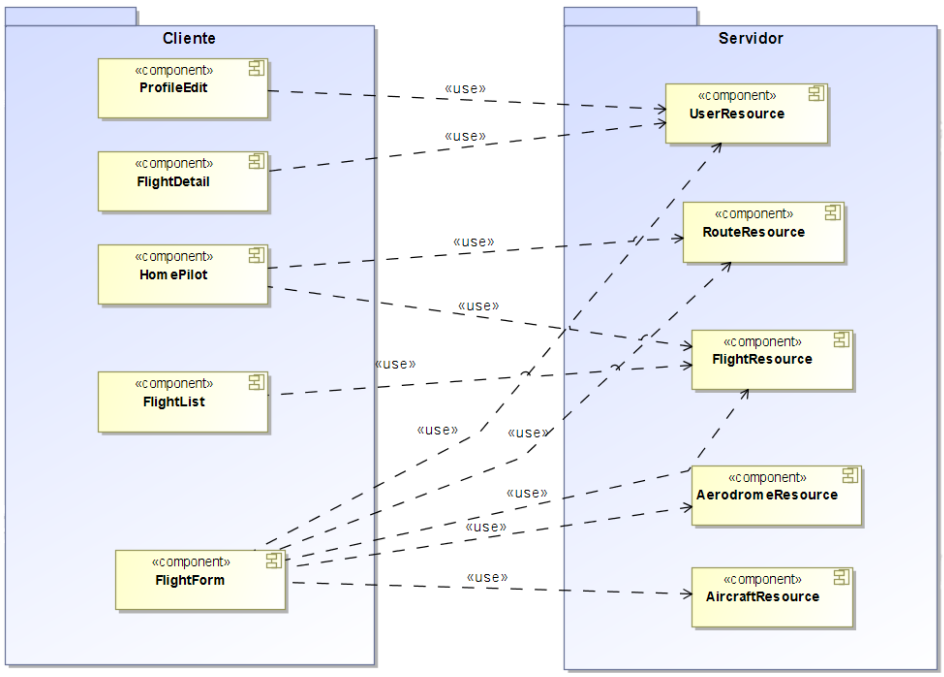


Figura 5.14: Comunicación entre los componentes de pilotos y el servidor (2).

- Por último, se muestra la comunicación para los componentes de **usuarios anónimos** y para los **componentes comunes**.

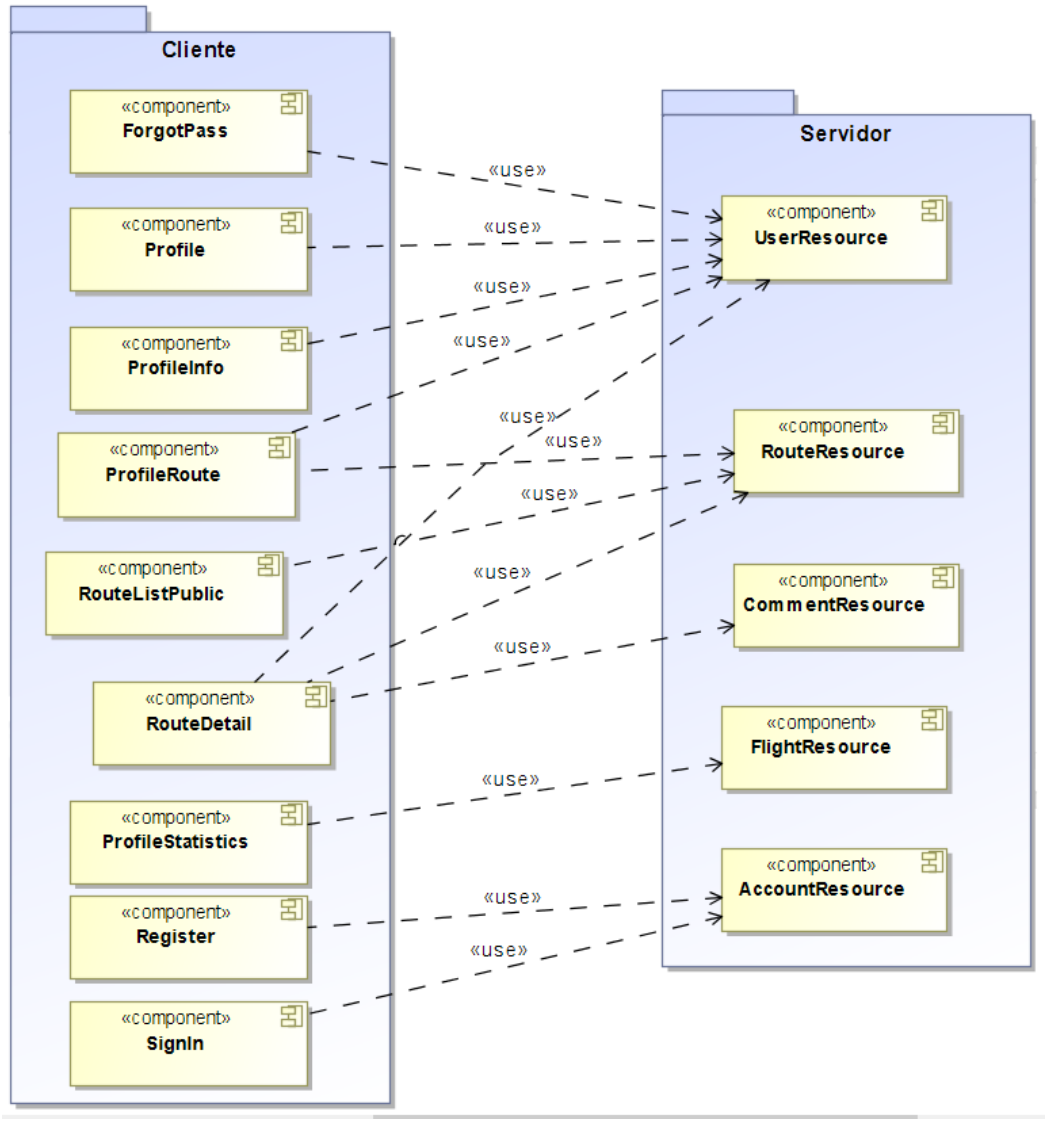


Figura 5.15: Comunicación entre los componentes de anónimos y el servidor.

5.2.3 Seguridad

Uno de los requisitos no funcionales descritos en la Sección 4.1.3 era que la aplicación debía asegurar la privacidad de los datos que los pilotos decidieran no compartir, así como no permitir que los usuarios normales ejerzan labores de administración. Por ello se ha decidido que la aplicación debe disponer de un sistema de autenticación que distinga usuarios anónimos de usuarios registrados y administradores. Para la identificación de los usuarios que

realizan las peticiones REST al servidor se ha optado por el estándar *JWT (JSON Web Token)*, el cual asigna un token a cada usuario que inicia sesión en la aplicación indicando los permisos que posee. Gracias a esto, el servidor no se verá obligado a guardar el estado de cada usuario, reduciendo el número de consultas a base de datos de forma considerable.

Además de esto, en la parte del cliente, cada vez que un usuario inicie sesión se guardará en el estado de ***Redux*** su información, la cuál se podrá enviar a cualquier componente en forma de propiedades. Con estos datos se podrá controlar el acceso a los componentes simplemente comprobando en las propiedades del usuario identificado en la aplicación el tipo de autorización que posee, ya sea de administrador o de piloto.

Capítulo 6

Implementación y pruebas

6.1 Introducción

En esta sección se mostrarán los algoritmos que se consideran más importantes para la aplicación por su complejidad o por el peso que tienen dentro de ésta. Además se indicarán los tipos de pruebas que se han realizado para comprobar el correcto funcionamiento del sistema.

6.2 Implementación

6.2.1 Creación de los recorridos de las rutas

Una de las características fundamentales de la aplicación es la creación de rutas de vuelo. Para tener acceso a esta funcionalidad, un usuario debe registrarse en la aplicación y acceder a la página de rutas. Una vez allí, tendrá la opción de crear una nueva ruta. Para ello deberá rellenar el formulario de creación de rutas, el cuál se encuentra en la URL */routes/new* y contiene los métodos indicados en el Cuadro 5.10. En dicho formulario el piloto, además de rellenar los campos necesarios para la creación de la ruta, deberá añadir un archivo GPX con la traza de su GPS. Dicho archivo contiene un documento XML con los distintos puntos por los que discurre la ruta marcados con la etiqueta *<trkpt />*, como el que se muestra en la Figura 6.1. Cada uno de esos puntos de ruta, además de contener información acerca de la fecha y hora y la altitud, indica en sus atributos las coordenadas exactas a las que se encuentra la aeronave en cierto momento en forma de latitud y longitud. Esas coordenadas serán las que se utilizarán en el método de creación del recorrido.

Una vez añadido el archivo GPX al formulario y habiendo rellenado los campos pertinentes, el usuario puede guardar la ruta en la aplicación. En este momento el sistema, además de hacer peticiones al servicio REST para guardar los datos de la ruta y sus imágenes, envía una petición específica (Figura 6.2) para enviar el fichero que contiene los datos GPS del recorrido de la ruta.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<gpx creator="Wikiloc - https://www.wikiloc.com" version="1.1" xmlns="http://www.topografix.com/GPX/1/1"
  <metadata>
    <name>Wikiloc - Barcelona (T2) - Ibiza mayo 2016</name>
    <time>2016-05-08T20:30:57Z</time>
  </metadata>
  <trk>
    <name>Barcelona (T2) - Ibiza mayo 2016</name>
    <cmt></cmt>
    <desc></desc>
    <trkseg>
      <trkpt lat="38.872981" lon="1.368936">
        <ele>2566.972</ele>
        <time>2016-05-08T05:50:23Z</time>
      </trkpt>
      <trkpt lat="38.873050" lon="1.368533">
        <ele>2555.386</ele>
        <time>2016-05-08T05:50:31Z</time>
      </trkpt>
      <trkpt lat="38.874115" lon="1.367612">
        <ele>2506.860</ele>
        <time>2016-05-08T05:51:07Z</time>
      </trkpt>
    </trkseg>
  </trk></gpx>
```

Figura 6.1: Ejemplo de un archivo GPX.

```
  HTTP.put(`routes/uploadfiles/${id}`, formData, {
    "Content-Type": "multipart/form-data"
  })
    .then(() => {
      notify.show("Route successfully created", "success", 3000);
      push("/routes");
    })
    .catch(() => {
      notify.show("Error uploading GPX file", "error", 3000);
    });
};
```

Figura 6.2: Petición HTTP PUT para almacenar el archivo GPX.

La URL de esta petición la lleva al controlador de rutas, concretamente al método *uploadRouteFiles*, que se muestra en la Figura 6.3. Este método sirve para los archivos citados y para las imágenes, por lo que lo primero que hace es comprobar que el archivo que le llega está en formato GPX, en cuyo caso llama a dos métodos del servicio de rutas, *store*, el cual se puede ver en la Figura 6.4 y lo que hace es comprobar que es un archivo válido y guardarlo en el sistema de ficheros y *parsePathFile*, que se explica en detalle a continuación.

En la Figura 6.5 se puede comprobar el algoritmo que analiza el archivo GPX adjunto a la ruta que se ha subido a la aplicación. Este método utiliza la librería de Java *JPX*, la cual es útil para leer y crear archivos de esta índole. Se usará el método *GPX.read* para analizar uno a uno todos los puntos de ruta incluidos en el archivo. De cada uno de estos puntos se extraerán sus coordenadas en forma de latitud y longitud y se incluirán en un array de coordenadas en forma de un objeto *Coordinate* incluido en el paquete *org.locationtech.jts*. Una vez completado el array con todos los puntos del fichero, se creará un objeto del tipo *LineString*, también incluido en el paquete *org.locationtech.jts*, el cual será el que se incluya en el atributo *path* de la ruta que se acaba de crear.

```
@PutMapping("/uploadfiles/{id}")
public void uploadRouteFiles(@PathVariable Long id, @ModelAttribute MultipartFile file, ModelMap modelMap)
        throws Exception{

    String filename = StringUtils.cleanPath(file.getOriginalFilename());
    String fileFormat = filename.substring(filename.length() - 4);

    if (fileFormat.equals(".gpx")) {
        routeService.store(file);
        routeService.parsePathFile(file, id);
    } else {
        modelMap.addAttribute("file", file);
        routeService.store(file);
    }

}
```

Figura 6.3: Método del controlador para el archivo GPX.

```
public void store(MultipartFile file) throws Exception {
    String filename = StringUtils.cleanPath(file.getOriginalFilename());

    try {
        if (file.isEmpty()) {
            throw new Exception("Failed to store empty file " + filename);
        }
        if (filename.contains("..")) {
            // This is a security check
            throw new Exception(
                    "Cannot store file with relative path outside current directory "
                            + filename);
        }
        try (InputStream inputStream = file.getInputStream()) {
            Files.copy(inputStream, this.location.resolve(filename),
                StandardCopyOption.REPLACE_EXISTING);
        }
    }
    catch (Exception e) {
        throw new Exception("Failed to store file " + filename, e);
    }
}
```

Figura 6.4: Método del servicio para guardar GPX en directorio.

```
@Transactional(readOnly = false)
public void parsePathFile(MultipartFile file, Long id) throws IOException {

    String filename = StringUtils.cleanPath(file.getOriginalFilename());
    String filepath = location.toString();

    List<Coordinate> coords = new ArrayList<>();

    GPX.read(filepath + "/" + filename).tracks()
    .flatMap(Track::segments)
    .flatMap(TrackSegment::points)
    .forEach(x -> coords.add(new Coordinate(x.getLatitude().doubleValue(), x.getLongitude().doubleValue())));

    Coordinate[] coordsArray = new Coordinate[coords.size()];
    coordsArray = coords.toArray(coordsArray);

    PrecisionModel pm = new PrecisionModel();
    LineString routePath = new GeometryFactory(pm, 4326).createLineString(coordsArray);

    Route bdRoute = routeDAO.findById(id);
    bdRoute.setPath(routePath);

    routeDAO.save(bdRoute);
}
```

Figura 6.5: Método del servicio para analizar el contenido del archivo GPX.

En el diagrama de secuencia de la Figura 6.6 se puede observar la interacción entre el cliente y las clases involucradas (controlador, servicio y acceso a datos) en la creación de rutas y del recorrido de éstas, además de la posterior recuperación para visualizarlo en la vista de detalle.

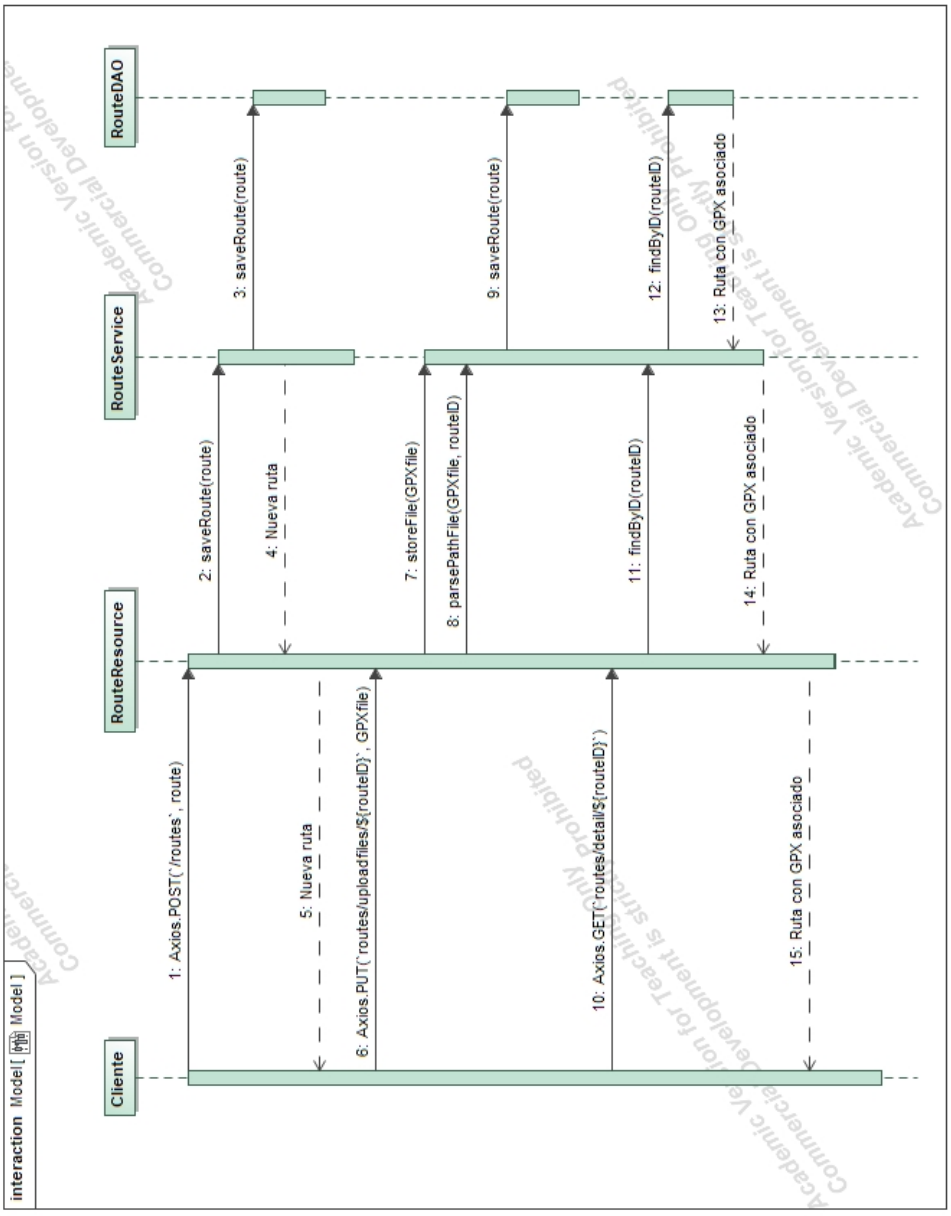


Figura 6.6: Diagrama de secuencia para la creación del recorrido de ruta.

6.2.2 Visualización de los recorridos de las rutas

En la aplicación existen dos maneras de visualizar los recorridos de las rutas, excluyendo la página principal: desde el detalle de la ruta o desde el detalle de un vuelo que haya seguido dicha ruta. Cada uno de ellos se muestra de forma distinta:

En la pantalla de **detalle de ruta**, la cual se muestra en la Figura 7.21, aparece el componente *MapRoute.jsx*, que muestra el recorrido de la ruta, las ubicaciones de los aeródromos implicados y las ubicaciones de las imágenes asociadas. Para ello, el componente *RouteDetail.jsx* realiza una petición GET al servidor para disponer de los atributos de la ruta que se quiere exponer. Una vez recibidas y almacenadas en el estado, llama al componente del mapa pasando como propiedades las coordenadas de los aeródromos de salida y llegada, las coordenadas de la ubicación de las imágenes asociadas y un array de coordenadas con el recorrido punto a punto de la ruta, almacenado de la forma que se describe en la Sección 6.2.1.

Una vez recibidos los parámetros necesarios, el componente *MapRoute.jsx* utiliza la librería *Leaflet* para crear un mapa en el que exponer dichas coordenadas. Primero se debe seleccionar una capa base escogida de la extensión *leaflet-providers* [24] para tener una base en la que cargar los diferentes elementos. A continuación se presenta el código necesario para su exposición:

- Para los **aeródromos**, se crearán dos marcadores, cada uno con su icono para poder distinguirlos. Al hacer clic sobre alguno de estos dos marcadores, se abrirá una ventana emergente que mostrará el nombre del aeródromo en cuestión.

```
this.takeoffAirport = L.marker(this.props.takeoffAirport, {
  icon: airportIcon
})
  .bindPopup(
    L.popup().setContent(
      "<center><p><b>" + this.props.departure.name + "</b></p></center>"
    )
  )
  .addTo(this.map);

this.landingAirport = L.marker(this.props.landingAirport, {
  icon: endIcon
})
  .bindPopup(
    L.popup().setContent(
      "<center><p><b>" + this.props.arrival.name + "</b></p></center>"
    )
  )
  .addTo(this.map);
```

Figura 6.7: Creación de marcadores para los aeródromos.

- Para las **imágenes**, se creará un marcador con el icono de una cámara de fotos para cada una de ellas. Al hacer clic en el marcador, se abrirá una ventana emergente con el título de la imagen y una miniatura de la misma.

```
this.props.images.map((image, index) => {
  let popup = L.popup();

  let imageMarker = L.marker(image.position.coordinates, {
    icon: cameraIcon,
    title: image.name,
    riseOnHover: true
  })
    .bindPopup(
      popup.setContent(
        "<center><h5><b>" +
          image.name +
          "</b></h5></center>" +
          "<img src='" +
          this.props.imagePaths[index] +
          "'height=200px width=250px</img>"
      )
    )
    .addTo(this.map);

  return imageMarker;
});
```

Figura 6.8: Creación de marcadores para las imágenes.

- El **recorrido de la ruta** se mostrará como una línea creada con los distintos puntos que contiene el array de coordenadas que recibe el componente.

```
this.pathLine = L.polyline(this.props.routePath, { color: "#a14655" }).addTo(
  this.map
);
```

Figura 6.9: Creación de linea de recorrido de ruta.

Las Figuras 6.10 y 6.11 muestran el mapa que se incluye en la vista de detalle de la ruta una vez se ejecuta el código indicado para cada elemento.

Figura 6.10: Mapa con el recorrido indicado por el archivo GPX.

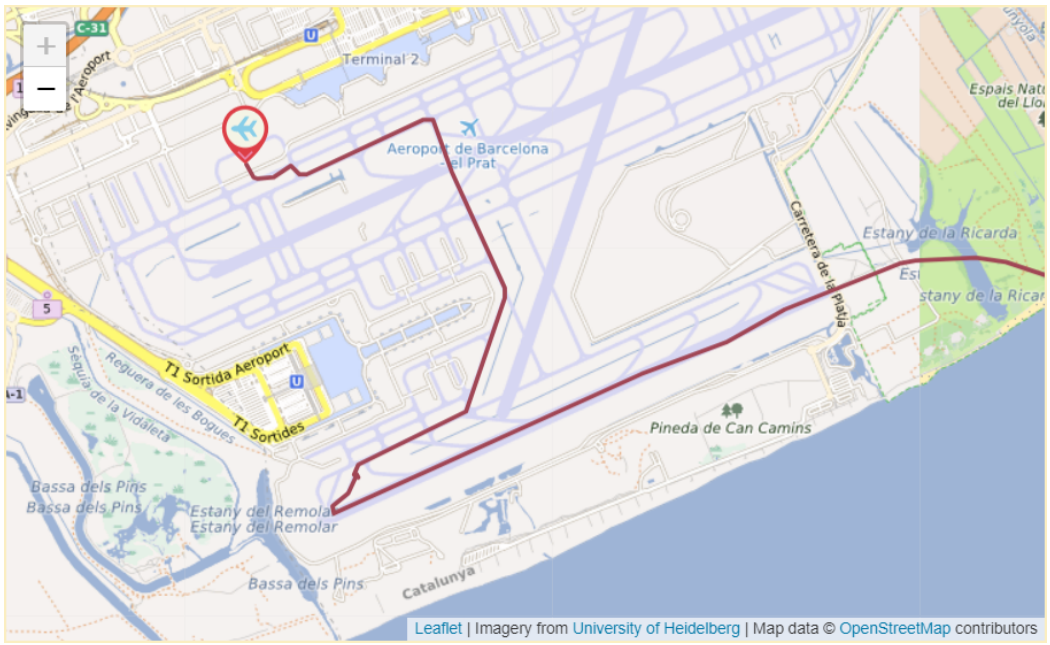


Figura 6.11: Mapa con el recorrido indicado por el archivo GPX (2).

Para la página de **detalle de vuelo**, que se muestra en la Figura 7.27, también se incluye un mapa, el cual mostrará el recorrido de la ruta en caso de que dicho vuelo siga una. La diferencia principal con el mapa del detalle de ruta es que éste utilizará una animación para dibujar el recorrido de la ruta una vez se cargue la página.

Primero, igual que en el caso anterior, se crean una capa base y los marcadores de los aeródromos. Para que se dibuje el recorrido de la ruta una vez cargada la página se ha utilizado la extensión *Leaflet motion* [25]. En el código mostrado en la Figura 6.12 se establecen los requisitos necesarios para la animación, como el color de la línea, el tiempo que va a tardar, etc.

```
  L.motion
    .polyline(
      this.props.polyline,
      {
        color: "#a14655"
      },
      {
        auto: true,
        duration: 6000,
        easing: L.Motion.Ease.easeInOutQuart
      },
      {
        removeOnEnd: false,
        icon: airportIcon
      }
    )
    .addTo(this.map);
}
```

Figura 6.12: Creación de la animación con el recorrido de la ruta.

6.3 Pruebas

6.3.1 Pruebas de integración

Se ha decidido comprobar el correcto funcionamiento de los distintos elementos que componen la parte del servidor de la aplicación a través de pruebas de integración. Con ellas se puede verificar que las capas se comunican correctamente entre ellas y devuelven el resultado esperado.

Spring-boot proporciona un módulo que permite crear pruebas de integración llamado

spring-boot-starter-test. Hemos añadido la dependencia de dicho módulo al fichero de configuración de Maven para poder trabajar con él.

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>
```

Figura 6.13: Dependencia de spring-boot-starter-test.

Se ha creado un fichero de test para probar cada uno de los controladores. Las clases se han anotado con *@RunWith* para permitir inyectar las dependencias que nos interesen y con *@SpringBootTest* para indicarle el nombre del fichero donde se inicializa la aplicación y el puerto en el que se ejecutará el entorno web ficticio para las pruebas.

Una vez ejecutadas las pruebas se ha comprobado el correcto funcionamiento de los controladores, como se puede comprobar en la Figura 6.14.

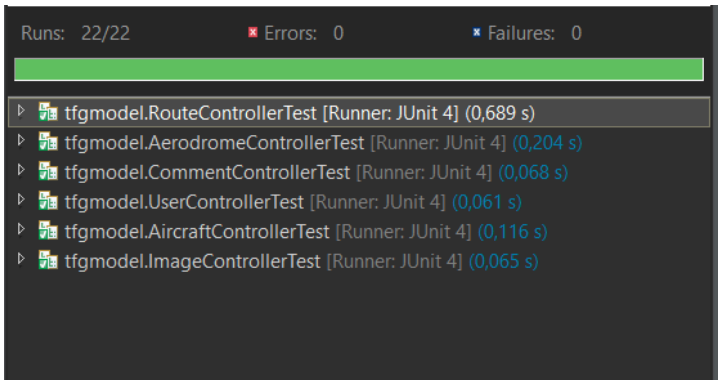


Figura 6.14: Resultados de las pruebas de integración.

6.3.2 Pruebas de aceptación

Se ha comprobado el correcto funcionamiento de la aplicación probando todos los casos de uso disponibles en la interfaz de usuario, combinando posibles resultados que podrían crear inconsistencias. Se ha intentado acceder a las páginas de las que no se dispone de los permisos necesarios, comprobando que la aplicación no permite el acceso, se han creado rutas con sus imágenes y asignándolas a vuelos para luego borrarlas y comprobar que todo funciona como debería, se ha experimentado con la creación de imágenes de rutas para verificar que cualquier combinación de creación y borrado no altera la integridad de la ruta. En cuanto a los perfiles de usuario, se ha procurado que se muestre la información correspondiente dependiendo de si se muestra el perfil del usuario autenticado o el de un usuario ajeno.

También se han probado los casos de uso del administrador comprobando que las aeronaves y aeródromos creados se muestran correctamente en los formularios de creación de vuelos y rutas y que el sistema no permite eliminar aquellos que están en uso. Para las labores de moderación se ha comprobado que los únicos usuarios que pueden eliminar los comentarios son el propietario de la ruta, el creador del comentario y los administradores.

Estas pruebas las ha realizado tanto el autor a lo largo del desarrollo, como los directores en las reuniones de revisión del *Sprint*.

Capítulo 7

Solución desarrollada

7.1 Introducción

En esté capítulo se mostrarán las características principales de la aplicación a través de una pequeña guía ilustrada dividida en tres partes: el acceso a la aplicación como usuario anónimo, la gestión de aeródromos y aeronaves por parte del administrador y finalmente las funcionalidades a las que pueden acceder los usuarios registrados.

7.2 Acceso a la aplicación

Un usuario que no se ha registrado puede acceder a funcionalidades limitadas en la aplicación. Habrá una página principal (Figura 7.1) en la que se le mostrarán las tres últimas rutas publicadas con un mapa en el que se indican sus recorridos y los aeródromos de salida y llegada de cada una. El usuario podrá acceder a la vista de detalle de cada una de esas rutas haciendo clic en la que le interese. En dicha vista de detalle se le limitarán considerablemente las interacciones con respecto a los usuarios registrados. Solamente podrá visualizar la información de la ruta, descargar el archivo GPX asociado a ella, ver los detalles de las imágenes adjuntas y visitar el perfil de usuario del creador de la ruta, en el que se le mostrará su información personal, las rutas que ha publicado, estadísticas de sus vuelos y sus listas de seguidores y seguidos. Además, desde el menú superior podrá acceder a la lista de rutas publicadas.

Para entrar a la aplicación como usuario registrado, se deberá haber creado previamente una cuenta. Para ello, existe una página de registro (Figura 7.2), accesible desde la página principal del usuario anónimo o desde el menú. Una vez creada, se podrán introducir las credenciales en la página de autenticación (Figura 7.3). Desde ésta, además, se podrá acceder a las pantallas de recuperación de la contraseña.

Figura 7.1: Página principal para usuarios anónimos.

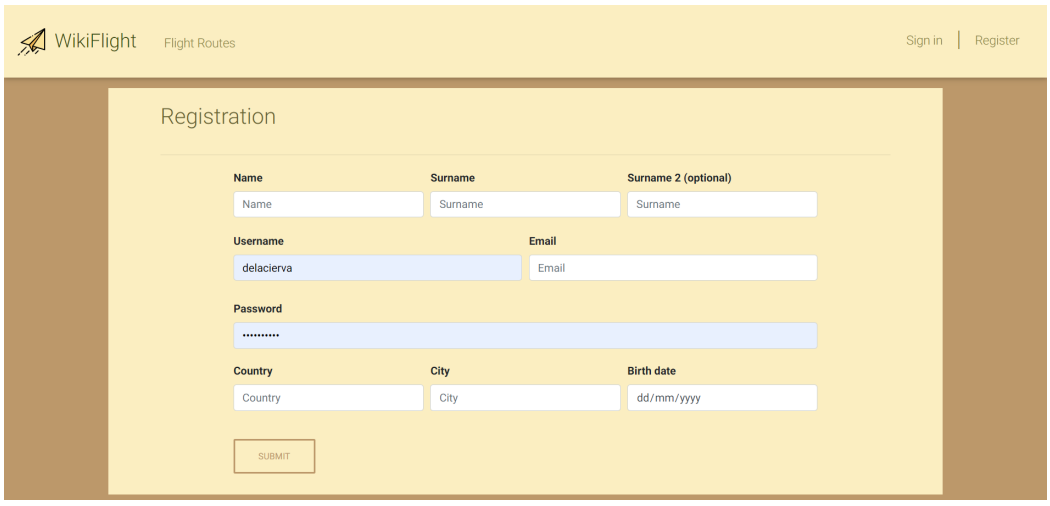


Figura 7.2: Página de registro.

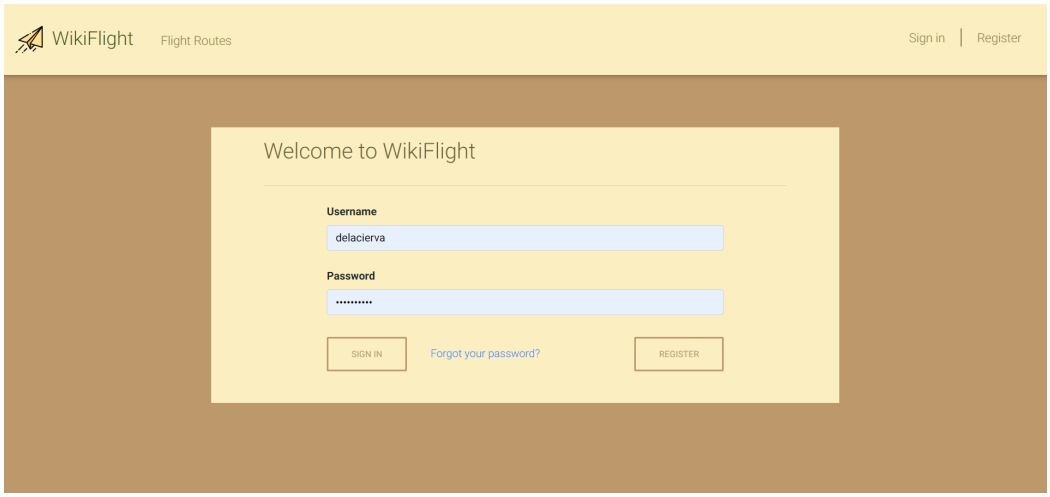


Figura 7.3: Página de autenticación.

7.3 Administración

Un administrador que se haya identificado como tal en la pantalla de autenticación tendrá tres funciones principales: gestión de aeronaves, gestión de aeródromos y moderación de comentarios. Para desarrollar esas funciones, existen pantallas dedicadas a cada una de ellas a las que se podrá acceder desde la página principal de los administradores (Figura 7.4) o desde el menú superior de la aplicación.

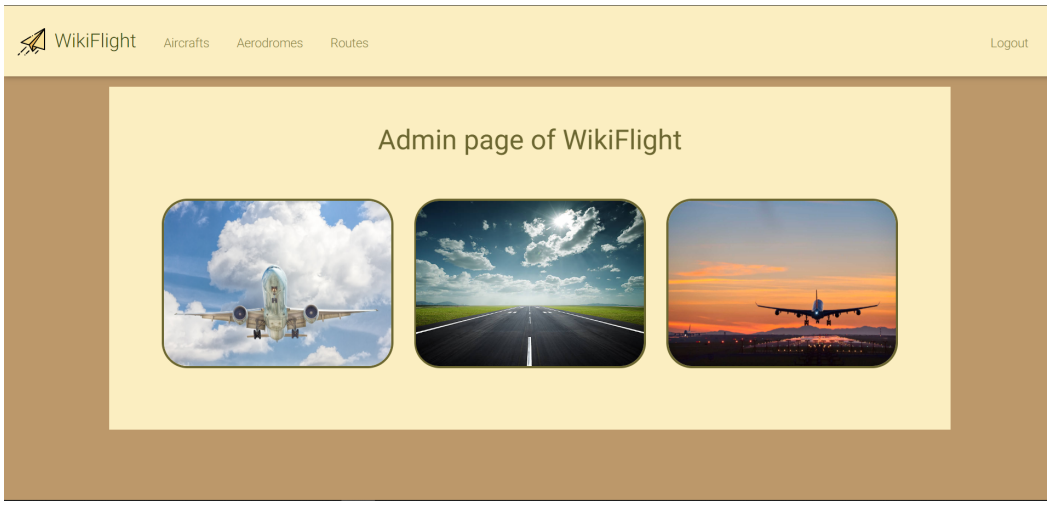


Figura 7.4: Página principal para administradores.

Para la **gestión de aeronaves** existe una pantalla con una lista en la que se muestran dichas aeronaves (Figura 7.5) indicando su fabricante y modelo. Al pasar el ratón por cada una, se mostrarán los botones para editarla y eliminarla. Además, en la parte superior de

la lista existe un botón para crear una nueva. En caso de pulsar ese botón o el de edición, el administrador será redirigido al formulario de aeronaves (Figura 7.6) para llevar a cabo dichas acciones.

Figura 7.5: Lista de aeronaves.

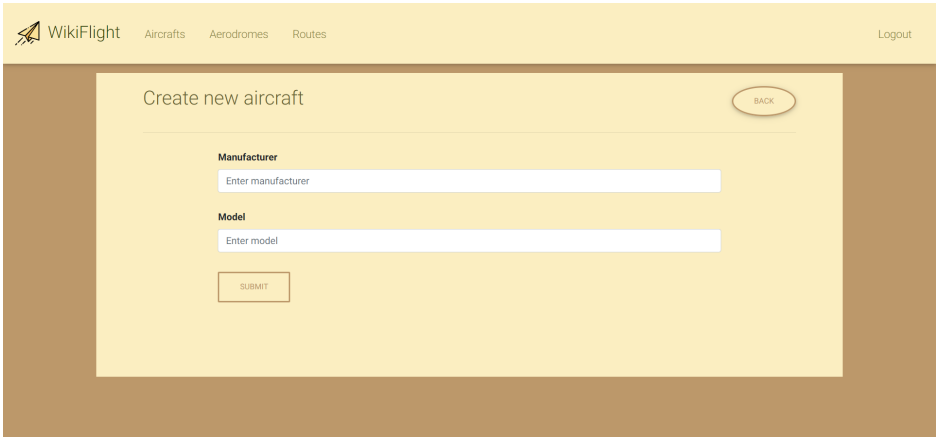


Figura 7.6: Formulario de creación y edición de aeronaves.

La **gestión de aeródromos** se llevará a cabo de la misma forma que la de aeronaves, con una lista (Figura 7.7) desde la que se podrá eliminar o editar cada uno de ellos y con un botón para crear uno nuevo. El formulario de aeródromos (Figura 7.8), además de incluir los campos pertinentes, cuenta con un mapa para indicar a mano la ubicación exacta donde se encuentra.

Por último, el administrador tendrá acceso a la lista de rutas publicadas (Figura 7.9), que se diferenciará de la lista a la que acceden el resto de usuarios porque en cada una de las rutas de la lista aparecerá indicado el número de comentarios escritos en ella. De esta forma, el administrador accederá a la vista de detalle de la ruta y tendrá la capacidad de eliminar los comentarios que considere inadecuados.

Figura 7.7: Lista de aeródromos.

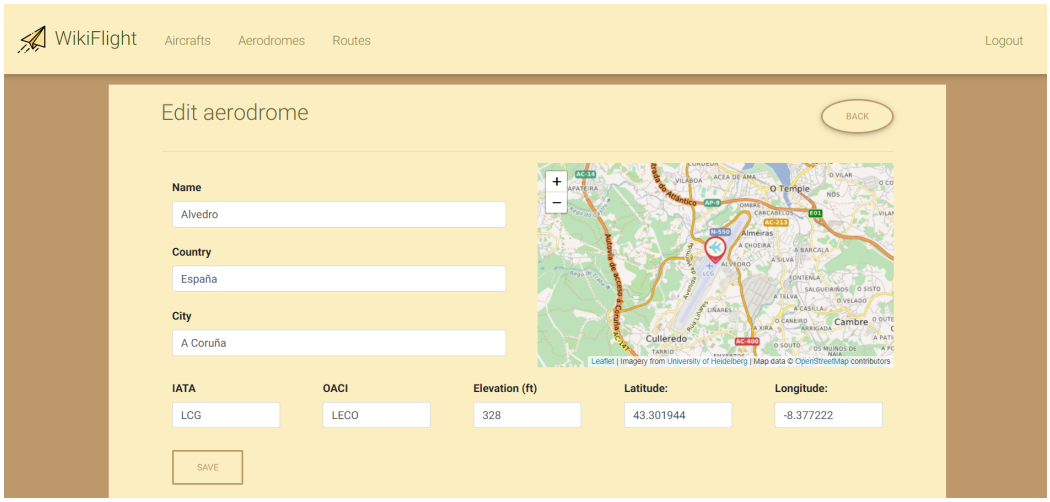


Figura 7.8: Formulario de creación y edición de aeródromos.

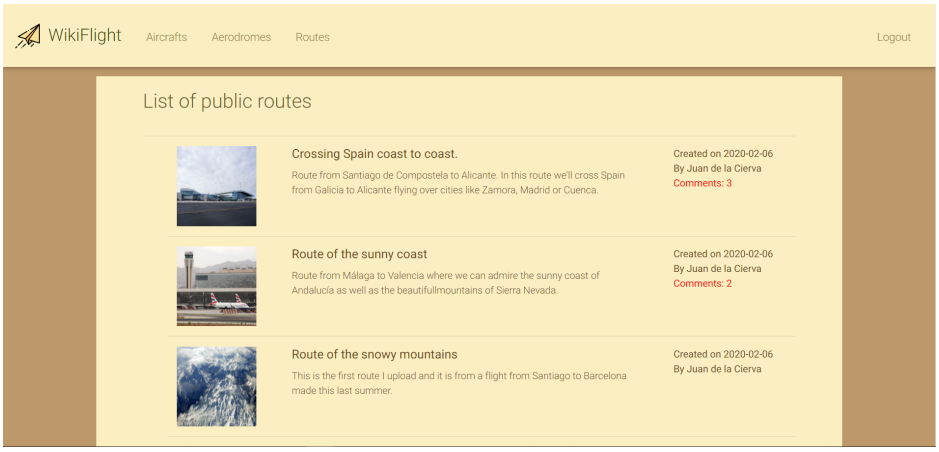


Figura 7.9: Lista de rutas para administradores.

7.4 Usuario Registrado

Un usuario registrado podrá disfrutar de la experiencia completa de la aplicación, aunque se le restringirá el acceso a las pantallas de administración.

Una vez se identifique, se le redirigirá a la **pantalla principal** de los pilotos, en la que, al igual que para los usuarios anónimos, se mostrará una lista con las tres últimas rutas publicadas acompañadas de una mapa que muestra los recorridos que siguen éstas (Figura 7.10). A diferencia de la pantalla principal de los usuarios anónimos, además de la lista de rutas, se mostrará una lista con los tres últimos vuelos realizados por el piloto (Figura 7.11), de los cuales se indicará su información básica. Haciendo clic en cualquier ruta o vuelo, se pasará a la pantalla de detalle de cada uno.

Figura 7.10: Página principal usuarios identificados.

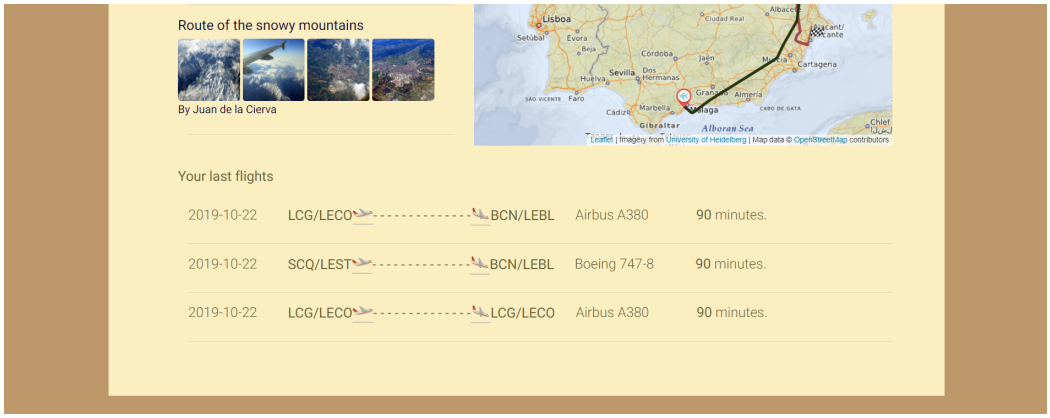


Figura 7.11: Página principal usuarios identificados (2).

Desde el menú superior de la página de pilotos, además de tener la posibilidad de ir a la página principal haciendo clic en el nombre de la página y cerrar sesión, se puede acceder a tres lugares: el perfil del piloto autenticado, la lista de rutas y el diario de vuelo personal del usuario identificado.

El **perfil de usuario** de un piloto cuenta con varias pestañas para cambiar la información que se muestra en el mismo, la cual variará en función de si el usuario visita su propio perfil o el de otros pilotos:

- La primera es la de **información personal**, donde se muestran el nombre, ciudad, país, fecha de nacimiento y fecha de registro de un piloto, además de una imagen de perfil. En caso de que se esté mostrando el perfil de usuario autenticado, aparecerá un botón para cambiar la imagen de perfil.

Figura 7.12: Perfil de usuario, pestaña de información.

- En la pestaña de **rutas** aparecerá una lista con las rutas que ha publicado el piloto del perfil con su nombre, descripción y algunas de sus imágenes asociadas. En caso de ser el perfil propio, se mostrarán todas las rutas creadas.

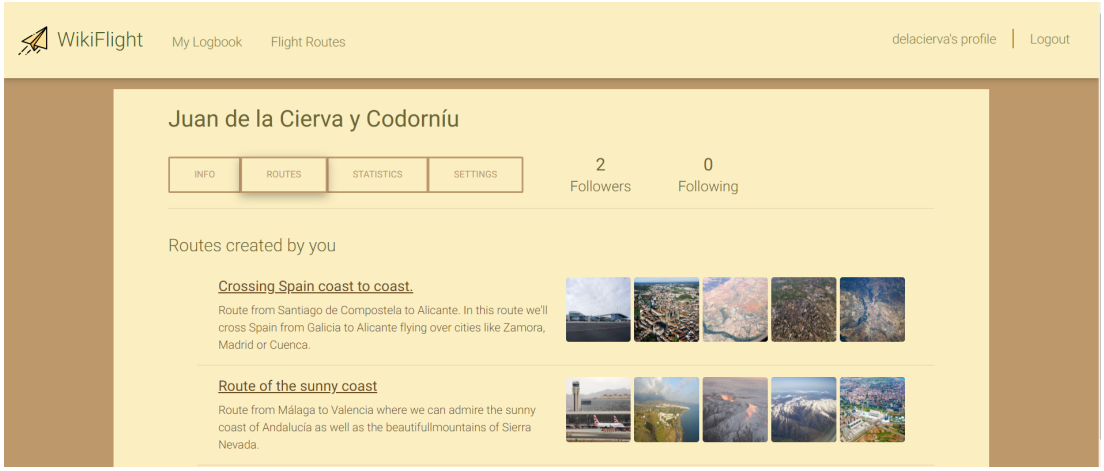


Figura 7.13: Perfil de usuario, pestaña de rutas.

- En las **estadísticas** aparecen algunos datos acerca de los vuelos que ha realizado el piloto, así como el número de rutas creadas y favoritas.

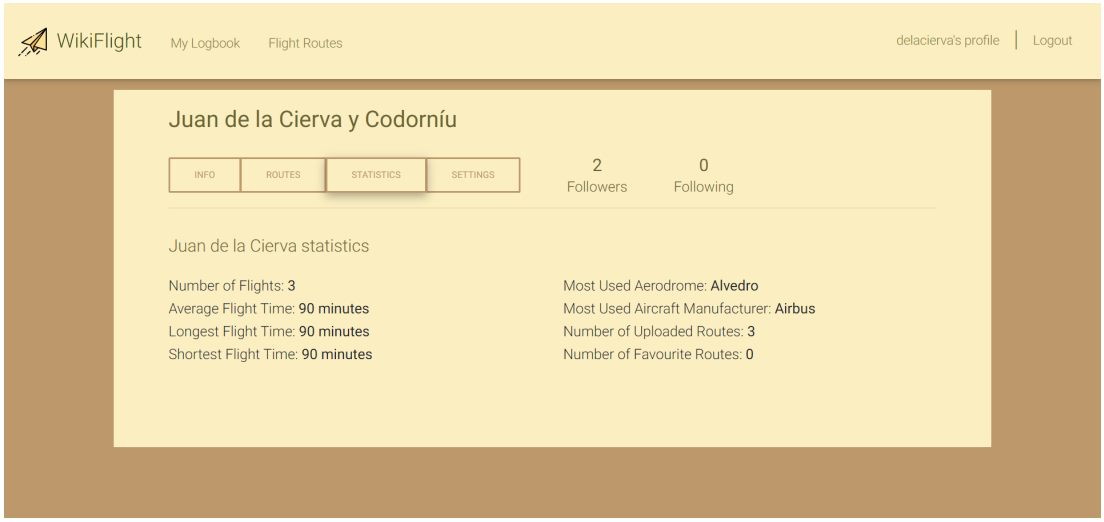


Figura 7.14: Perfil de usuario, pestaña de estadísticas.

- Si el perfil que se muestra es el del piloto identificado en la aplicación, aparecerá una pestaña de **ajustes** para modificar la información personal del usuario y la contraseña.
- Al visitar el perfil de otros pilotos aparecerá un botón que nos permite añadirlos o eliminarlos de nuestras listas de **seguidores y seguidos**

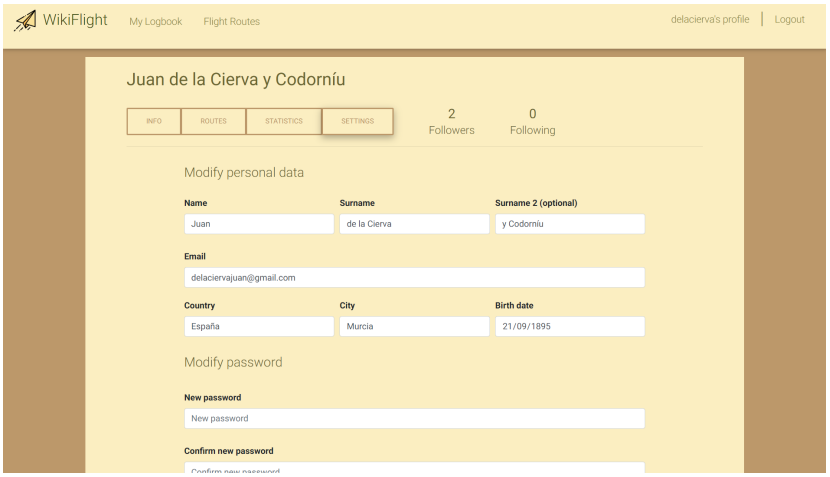


Figura 7.15: Perfil de usuario, pestaña de ajustes.

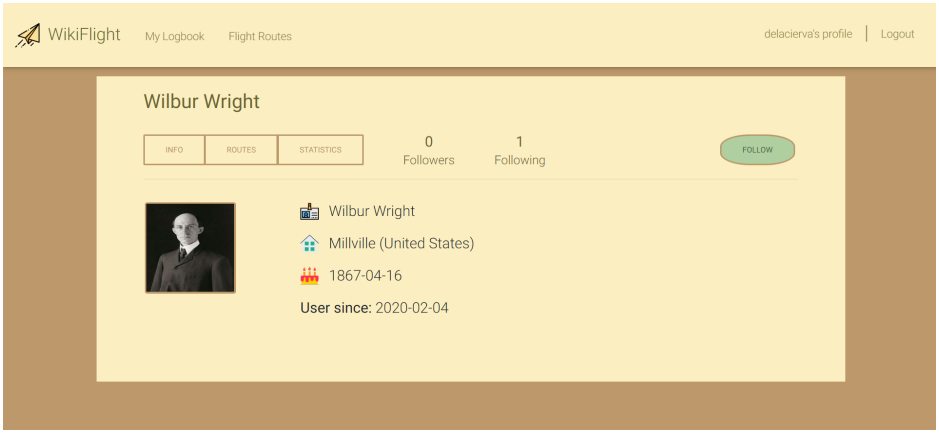


Figura 7.16: Visita al perfil de otro usuario.

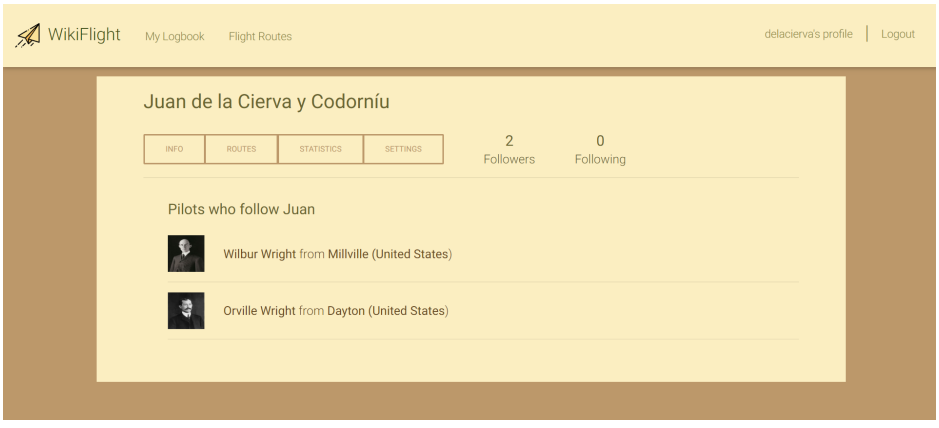


Figura 7.17: Perfil de usuario, pestaña de seguidores.

Las **rutas** aparecerán ordenadas por orden cronológico en la pantalla de lista de rutas (Figura 7.18), la cual contiene tres pestañas para filtrarlas: una lista de rutas públicas, otra con las listas creadas por el usuario identificado y una última con las rutas favoritas del piloto identificado. Además, en la parte superior de la lista existirá un botón para crear una nueva ruta, el cual redirigirá al usuario al formulario de creación y edición de rutas (Figura 7.19). En este formulario se podrán cubrir los campos necesarios para la creación o edición de una ruta, además de subir el archivo GPX asociado a ésta.

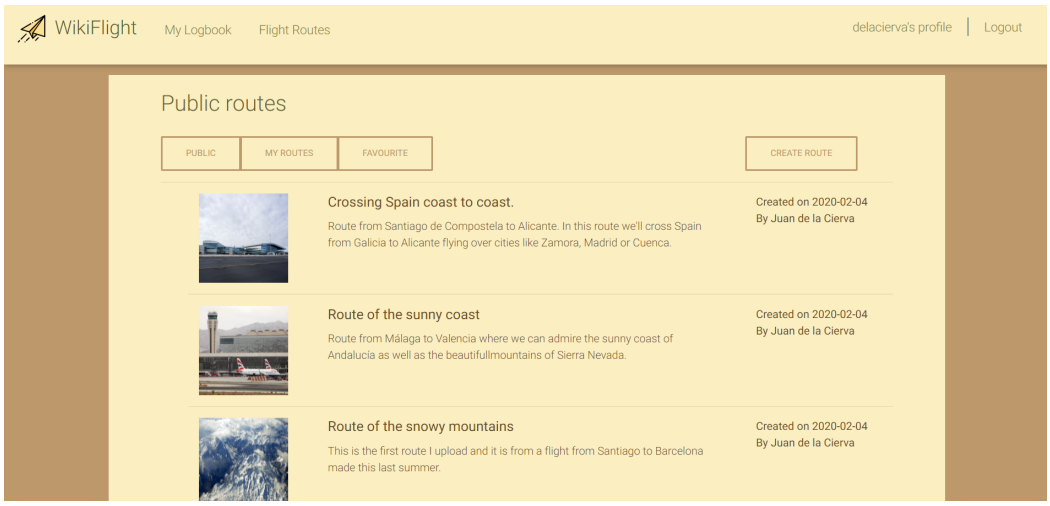


Figura 7.18: Listas de rutas.

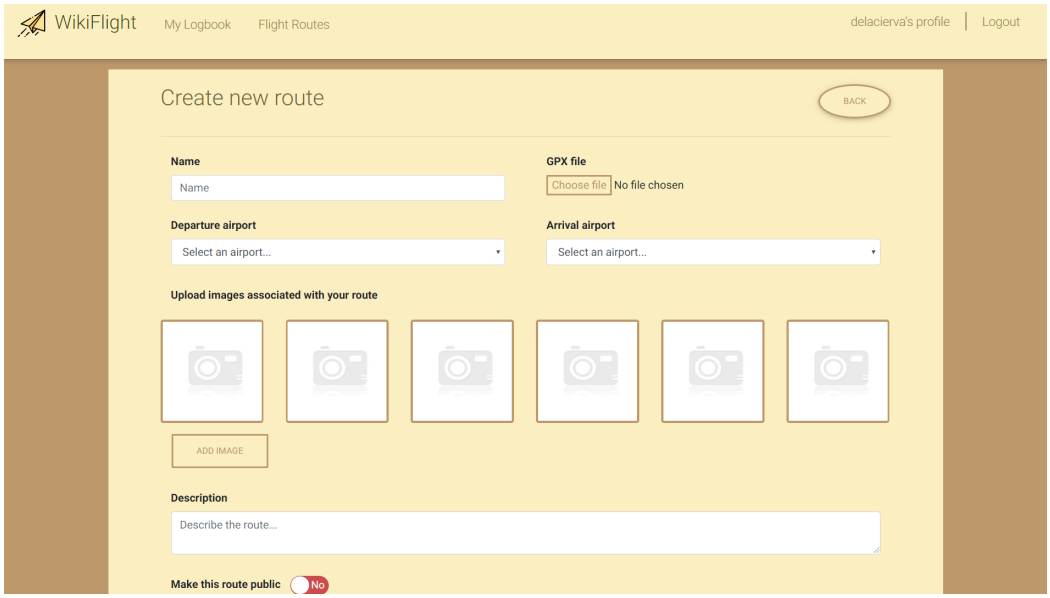


Figura 7.19: Formulario de creación y edición de rutas.

En el formulario también se tendrá la posibilidad de adjuntar imágenes a las rutas. Al pulsar el botón de añadir imagen en dicho formulario, se abrirá una ventana modal para subir la imagen deseada e indicar en un mapa su ubicación (Figura 7.20).

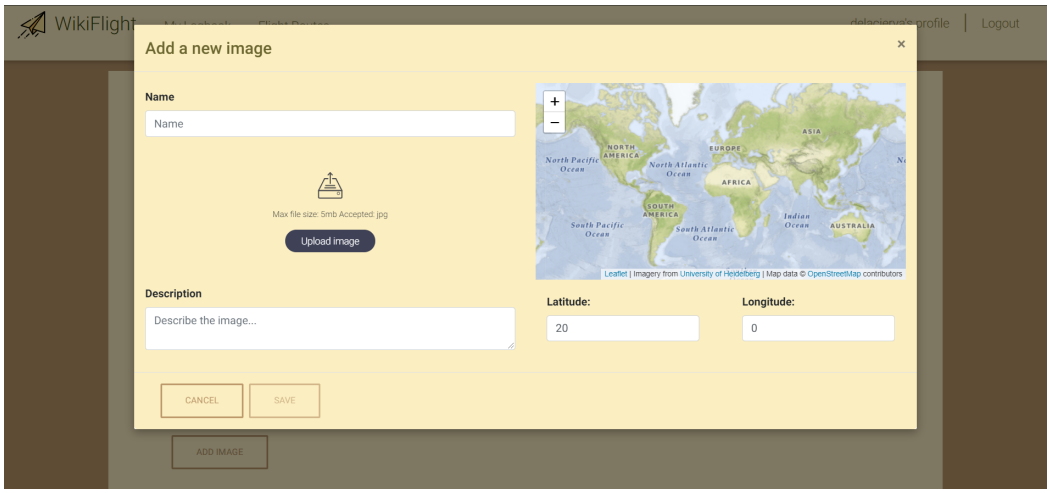


Figura 7.20: Formulario para adjuntar una imagen a una ruta.

Una vez creada una ruta, se podrá visitar su página de detalle (Figura 7.21). Aquí aparecerá la información indicada en la creación de la ruta, un mapa mostrando el recorrido y la ubicación de las imágenes y una miniatura de las imágenes en la que se puede hacer clic para verlas en grande y comprobar el título y la descripción (Figura 7.22). En la parte inferior de la página de detalle (Figura 7.23) aparece la descripción de la ruta y la sección de comentarios, donde cualquier piloto identificado podrá escribir una opinión acerca de la ruta.

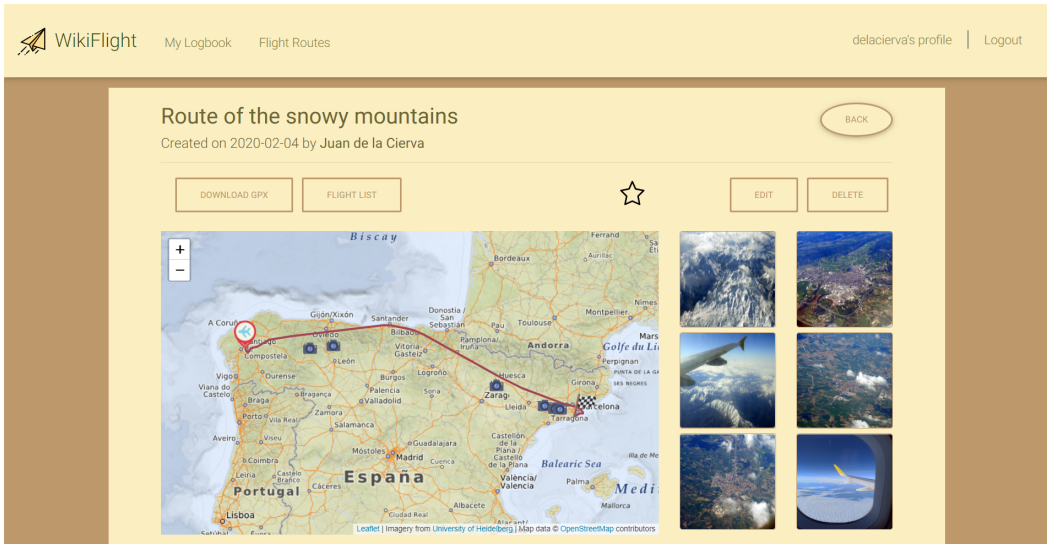


Figura 7.21: Detalle de ruta.

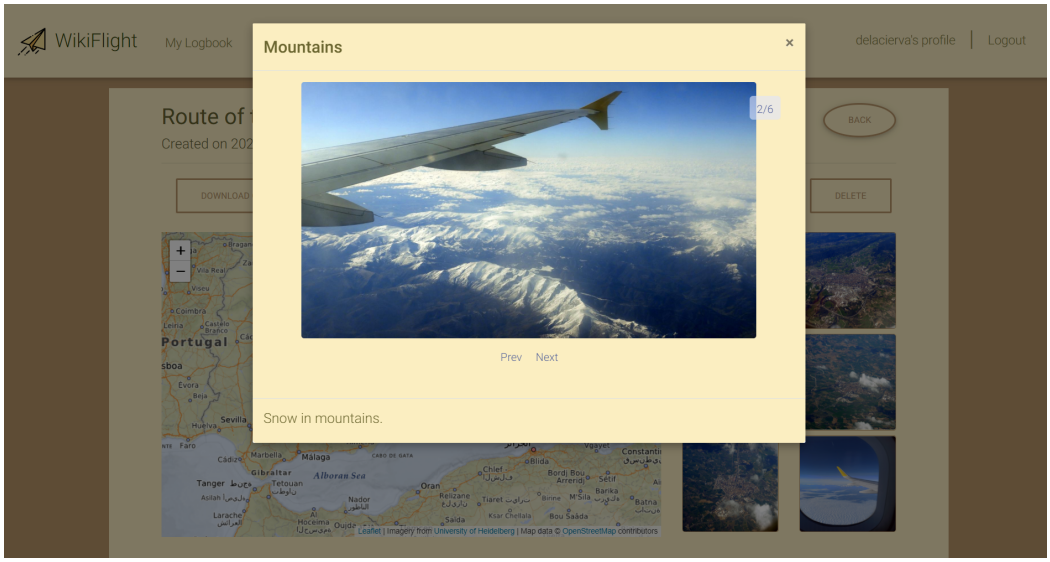


Figura 7.22: Detalle de imagen asociada a una ruta.

Figura 7.23: Detalle de ruta (2).

El **diario de vuelo** (Figura 7.24) consta de una tabla en la que se reflejan las entradas del diario que ha creado el piloto. En ellas se puede ver la información básica del vuelo, además de existir botones para editar la entrada, borrarla o verla en detalle. En la parte superior hay un botón para crear una nueva entrada y otro para descargar el diario en un archivo con formato CSV.

En caso de clicar el botón de crear una nueva entrada de vuelo o el de modificar una existente, se redirigirá al usuario al **formulario de vuelos** (Figuras 7.25 y 7.26). En caso de editar un vuelo, los campos aparecerán inicializados con los valores actuales.

Por último, al acceder al **detalle de una entrada del diario**, se le mostrará al usuario la información completa del vuelo y un mapa indicando su recorrido (Figura 7.27).

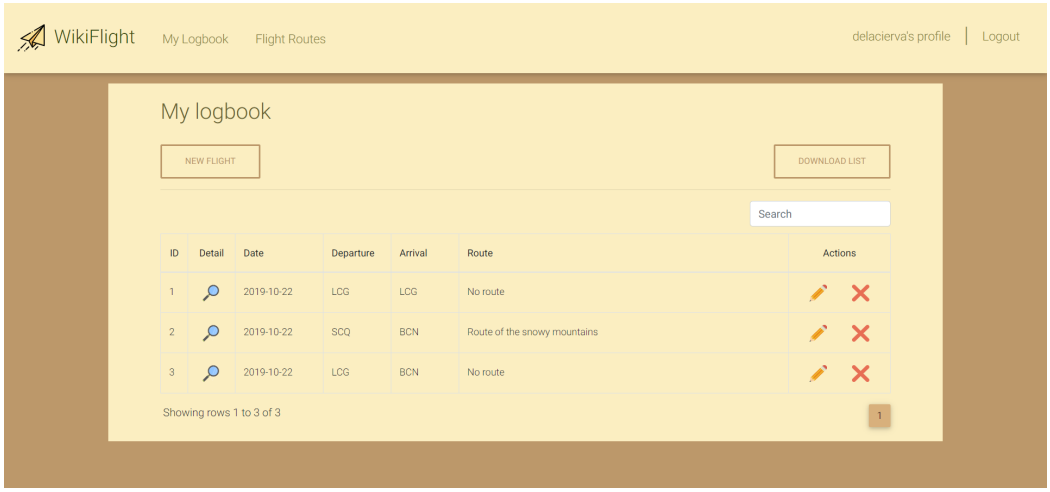


Figura 7.24: Entradas del diario de vuelo.

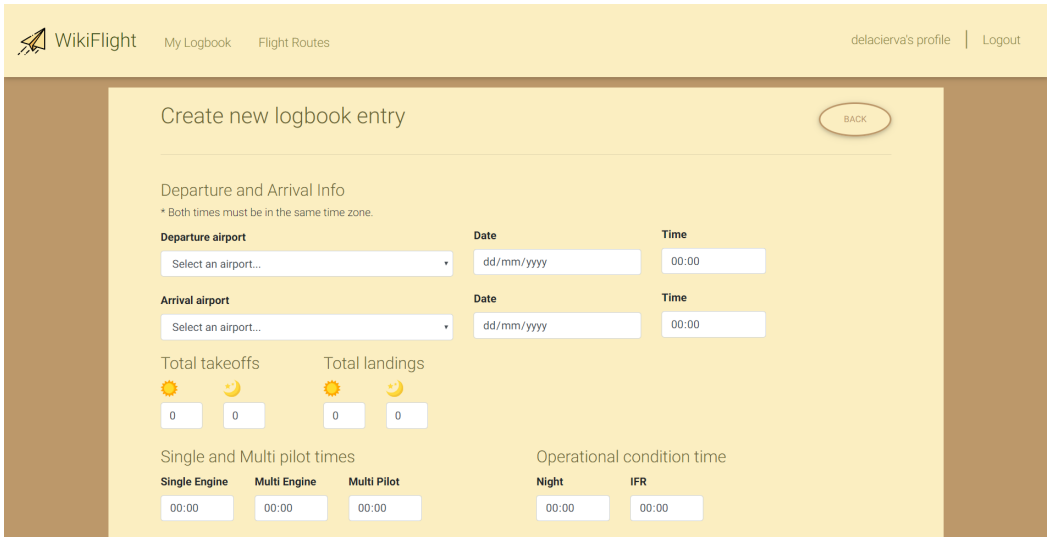


Figura 7.25: Formulario para la creación y edición de vuelos.

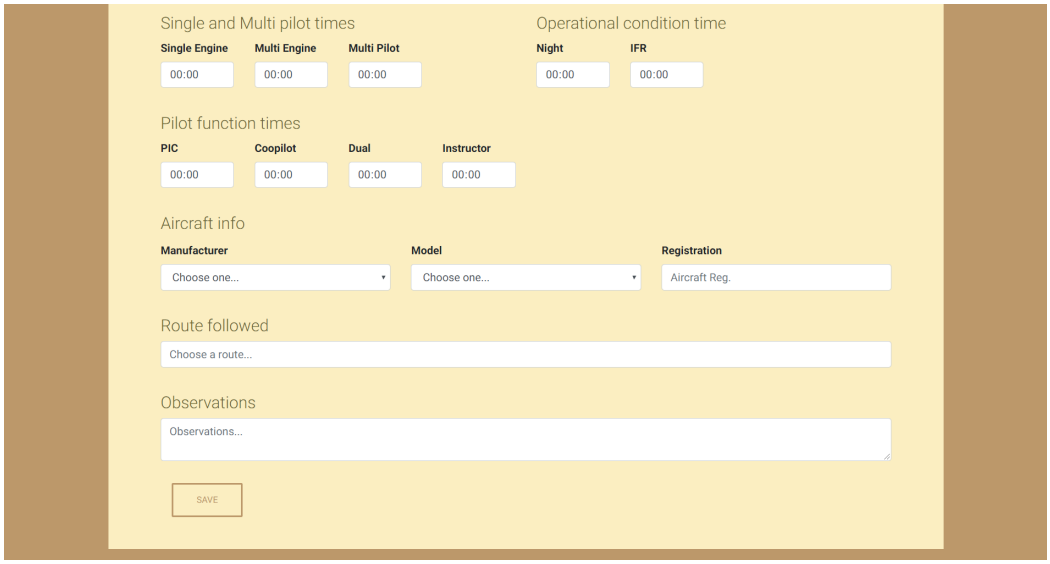


Figura 7.26: Formulario para la creación y edición de vuelos (2).

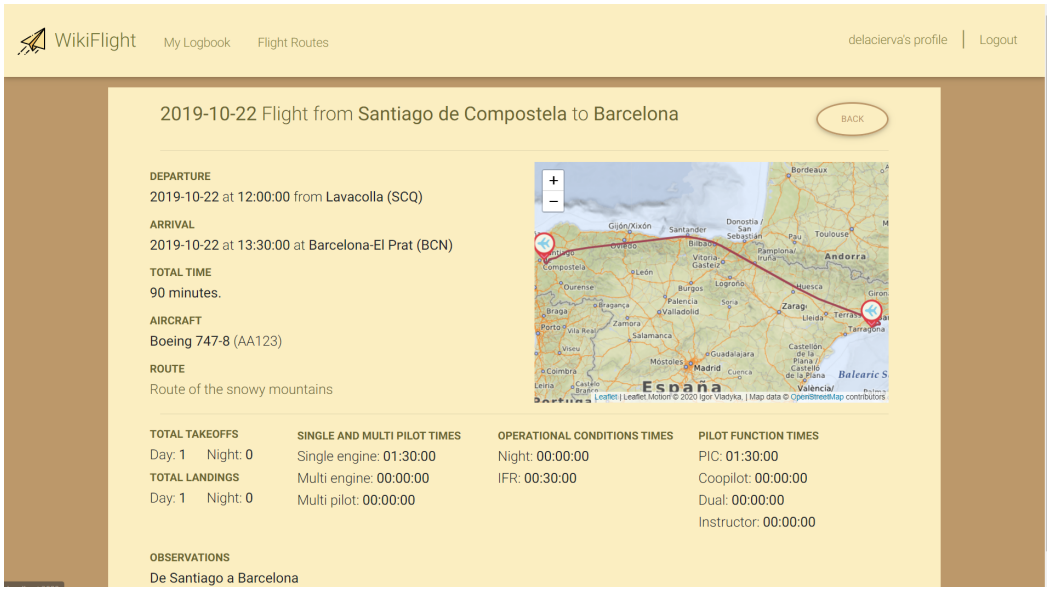


Figura 7.27: Detalle de una entrada del diario de vuelo.

Capítulo 8

Conclusiones y trabajo futuro

8.1 Conclusiones

Una vez finalizado el desarrollo del proyecto se han sacado las siguientes conclusiones:

- En lineas generales, se han cumplido los objetivos planteados al inicio de la planificación. La aplicación web que se ha creado obedece a las características descritas en un principio.

- Se ha creado un entorno web intuitivo, agradable y seguro para que el usuario se sienta cómodo navegando por la aplicación.

- Las funcionalidades principales de la aplicación (creación de rutas de vuelo y diario de vuelo del piloto) han sido implementadas de forma que cualquier usuario las pueda utilizar de forma sencilla y cómoda, y podrían considerarse de gran utilidad en un entorno real.

- Se han utilizado herramientas y tecnologías actuales para la realización de este proyecto, adquiriendo nuevos conocimientos tanto en la parte del *back-end*, trabajando con Java, Hibernate y Spring como en la parte del cliente, utilizando React y Leaflet, y se han ampliado los conocimientos adquiridos en la carrera.

- Ha tomado mucha importancia la metodología utilizada debido a la gran carga de trabajo que acarrea un proyecto de esta envergadura. El desarrollo por iteraciones ha permitido aligerar las distintas fases del trabajo y la resolución de los problemas que han ido surgiendo a lo largo del desarrollo.

8.2 Trabajo futuro

A pesar de que la aplicación cumple los objetivos descritos en un principio, hay algunos aspectos que podrían complementar la oferta de funcionalidades que tiene:

- Un **sistema de notificaciones** para que el usuario esté al día de la actividad de sus seguidores, así como de las interacciones que se van produciendo en sus rutas, ya sean nuevos comentarios o que algún piloto las haya añadido a sus listas de favoritas. También se podría producir comunicación entre administradores y usuarios en caso de que los primeros hayan eliminado un comentario de algún piloto en tareas de moderación.
- Sería interesante que se pudieran compartir con los seguidores un **vuelo en tiempo real**. Para ello el piloto podría programar un vuelo a una hora determinada y permitir al resto de usuarios que lo sigan en tiempo real en un mapa.
- Inclusión de **vídeos en las rutas**. Una de las características de las rutas es que se pueden adjuntar imágenes para ilustrar el recorrido. Una buena idea sería permitir incluir también vídeos, ya que la transmisión de vídeo en tiempo real sería complicada en zonas de gran altitud por la falta de cobertura.

Bibliografía

[1] "Página web de Wikiloc." [Online]. Available: https://es.wikiloc.com/

[2] "Página web de FlyLog.io." [Online]. Available: https://www.flylog.io/

[3] "Hibernate ORM." [Online]. Available: https://hibernate.org/orm/

[4] "Hibernate Spatial." [Online]. Available: https://docs.jboss.org/hibernate/orm/5.2/userguide/html_single/Hibernate_User_Guide.html#spatial

[5] "Spring Framework." [Online]. Available: https://spring.io/

[6] "PostgreSQL." [Online]. Available: https://www.postgresql.org/

[7] "PostGIS." [Online]. Available: https://postgis.net/

[8] "NodeJS." [Online]. Available: https://nodejs.org/es/

[9] "JSON Web Tokens." [Online]. Available: https://jwt.io/

[10] "Componentes de React Bootstrap." [Online]. Available: https://react-bootstrap.github.io/components/alerts/

[11] "Documentación de React." [Online]. Available: https://es.reactjs.org/docs/getting-started.html

[12] Valentino Gagliardi, "React Redux Tutorial for Beginners: The Definitive Guide," 2019. [Online]. Available: https://www.valentinog.com/blog/redux/

[13] "axios - Promise based HTTP client for the browser and node.js," 2019. [Online]. Available: https://github.com/axios/axios

[14] Vladimir Agafonkin, "Página web de Leaflet," 2019. [Online]. Available: https://leafletjs.com/

[15] Dan Foster, "GPX: the GPS Exchange Format." [Online]. Available: https://www.topografix.com/gpx.asp

[16] Ken Schwaber, Jeff Sutherland, "The Scrum Guide," 2017. [Online]. Available: https://www.scrumguides.org/scrum-guide.html

[17] "Eclipse IDE." [Online]. Available: https://www.eclipse.org/downloads/

[18] "Visual Studio Code." [Online]. Available: https://code.visualstudio.com/

[19] "MagicDraw." [Online]. Available: https://www.nomagic.com/products/magicdraw

[20] "Balsamiq Mockups." [Online]. Available: https://balsamiq.com/wireframes/

[21] "Draw.io." [Online]. Available: https://www.draw.io/

[22] "Node Package Manager." [Online]. Available: https://www.npmjs.com/

[23] "Overleaf." [Online]. Available: https://www.overleaf.com/project

[24] Stefan Seelmann, "Leaflet Providers," 2019. [Online]. Available: https://leaflet-extras.github.io/leaflet-providers/preview/

[25] Igor Vladyka, "Animation Plugin for Leaflet.js," 2019. [Online]. Available: https://github.com/Igor-Vladyka/leaflet.motion

[26] "Spring Boot." [Online]. Available: https://spring.io/projects/spring-boot

[27] "Spring Security." [Online]. Available: https://spring.io/projects/spring-security

[28] Lokesh Gupta, "Spring boot integration test example." [Online]. Available: https://howtodoinjava.com/spring-boot2/testing/spring-integration-testing/

[29] "Guía de inicio rápido de PostGIS." [Online]. Available: https://live.osgeo.org/es/quickstart/postgis_quickstart.html

[30] "Documentación de GeoTools." [Online]. Available: https://docs.geotools.org/

[31] Andrew Cherniavskii, "Using Leaflet in React apps," 2017. [Online]. Available: https://cherniavskii.com/using-leaflet-in-react-apps/

[32] Kamil Wojewski, "JWT Authentication in a React-Redux App," 2017. [Online]. Available: https://cherniavskii.com/using-leaflet-in-react-apps/

[33] Eugen Paraschiv, "Spring Security – Reset Your Password," 2019. [Online]. Available: https://www.baeldung.com/spring-security-registration-i-forgot-my-password/

[34] "File Upload With Spring MVC," 2019. [Online]. Available: https://www.baeldung.com/spring-file-upload

[35] Robin Wieruch, "How to use SVG Icons as React Components," 2019. [Online]. Available: https://www.robinwieruch.de/react-svg-icon-components

[36] "Flaticons - Free Vector Icons." [Online]. Available: https://www.flaticon.com/

[37] Franz Wilhelmstötter, "JPX - Java GPX library," 2019. [Online]. Available: https://github.com/jenetics/jpx

[38] Captain Joe, "How do you fill out your PILOT LOGBOOK," 2017. [Online]. Available: https://www.youtube.com/watch?v=Y6bU5XpG3n4

[39] Eric Giovanola, "React typeahead with Bootstrap styling ," 2019. [Online]. Available: https://github.com/ericgio/react-bootstrap-typeahead

[40] Markus Englund, "React Toggle Switch ," 2019. [Online]. Available: https://github.com/markusenglund/react-switch

[41] Max Marinich, "React Alice Carousel," 2019. [Online]. Available: https://github.com/maxmarinich/react-alice-carousel

[42] Allen Fang, "React Bootstrap Table 2," 2019. [Online]. Available: https://github.com/react-bootstrap-table/react-bootstrap-table2

[43] Jan Doležel, "React CSV Downloader," 2019. [Online]. Available: https://github.com/dolezel/react-csv-downloader

[44] Mohan Upadhyay, "React Loader Spinner," 2019. [Online]. Available: https://github.com/mhnpd/react-loader-spinner

[45] Jake Hartnell, "Images uploader UI component," 2019. [Online]. Available: https://github.com/jakehartnell/react-images-upload

[46] Jesús Otero Gómez, "Toast Notifications for React.js," 2019. [Online]. Available: https://github.com/jesusoterogomez/react-notify-toast

[47] "GitLab." [Online]. Available: https://gitlab.lbd.org.es/users/sign_in

Apéndices

Apéndice A

Contenido del CD

El CD contiene los siguientes ficheros:

- **EstevezAlvarezPabloTFG2020.pdf:** La memoria del trabajo en formato PDF.
- **EstevezAlvarezPabloTFG2020resumo.pdf:** El resumen del trabajo en formato PDF.
- **EstevezAlvarezPabloTFG2020anexo.zip:** Incluye por se parado el código fuente del servidor (**tfgmodel**) y del cliente (**tfgclient**).

Apéndice B

Prototipos de pantalla

A continuación se presentan los distintos prototipos de pantalla creados en la fase de análisis preliminar del proyecto para tener una idea de cómo debería ser la interfaz de usuario y los casos de uso presentes en cada pantalla.

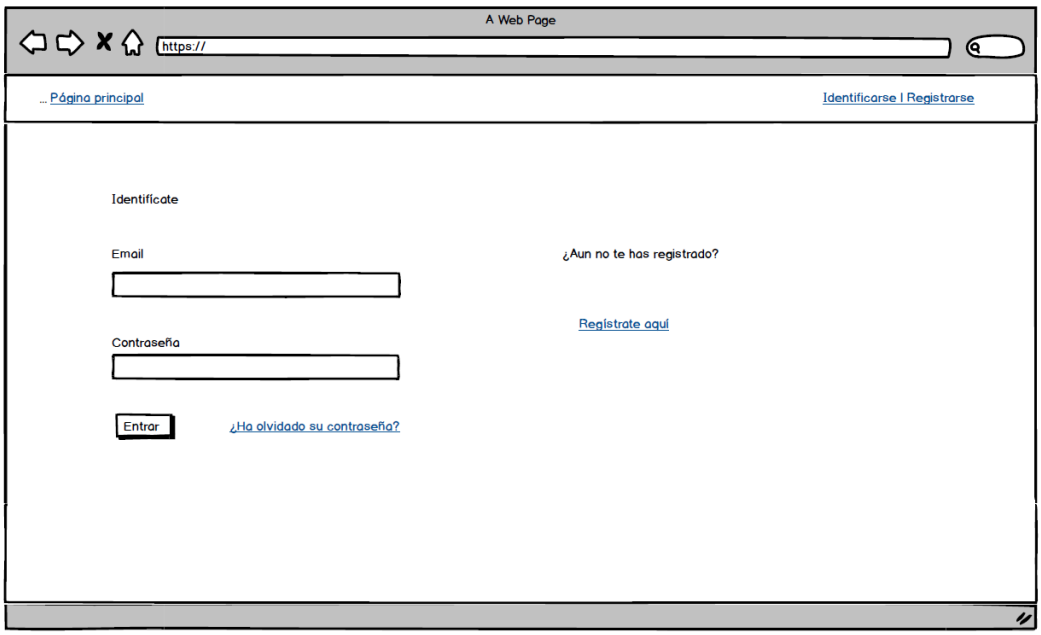


Figura B.1: Pantalla de autenticación.

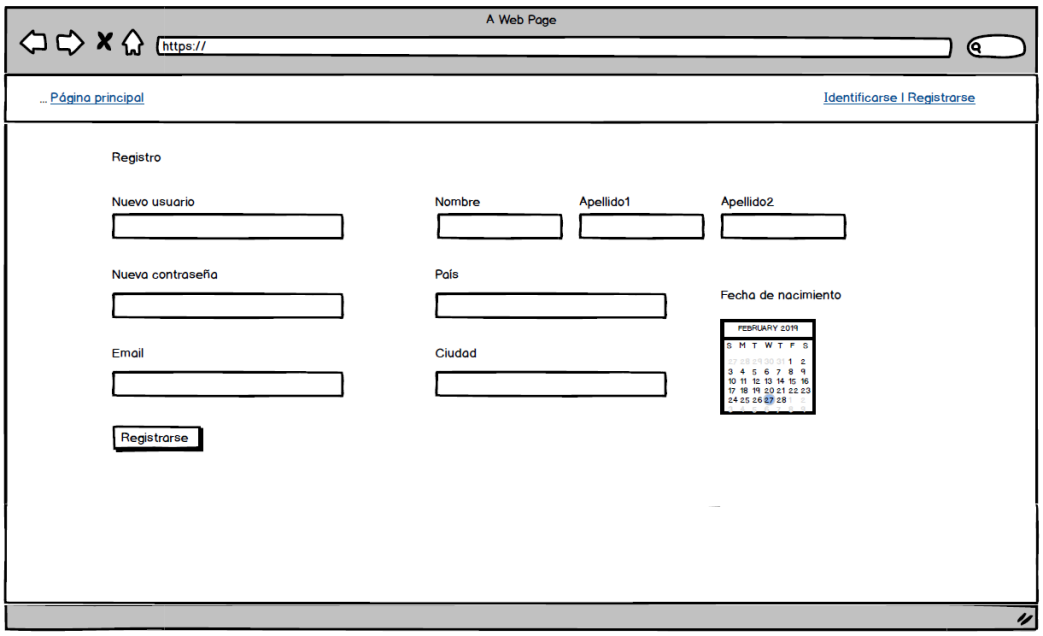


Figura B.2: Pantalla de registro.

Figura B.3: Pantalla principal para usuarios anónimos.

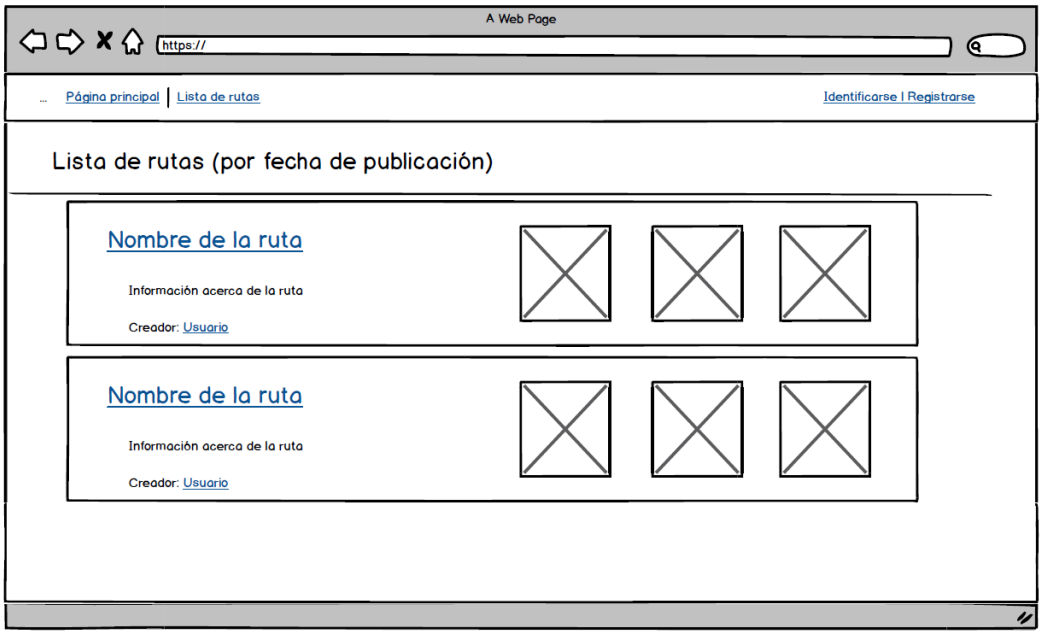


Figura B.4: Pantalla de lista de rutas públicas.

Figura B.5: Pantalla principal para usuarios registrados.

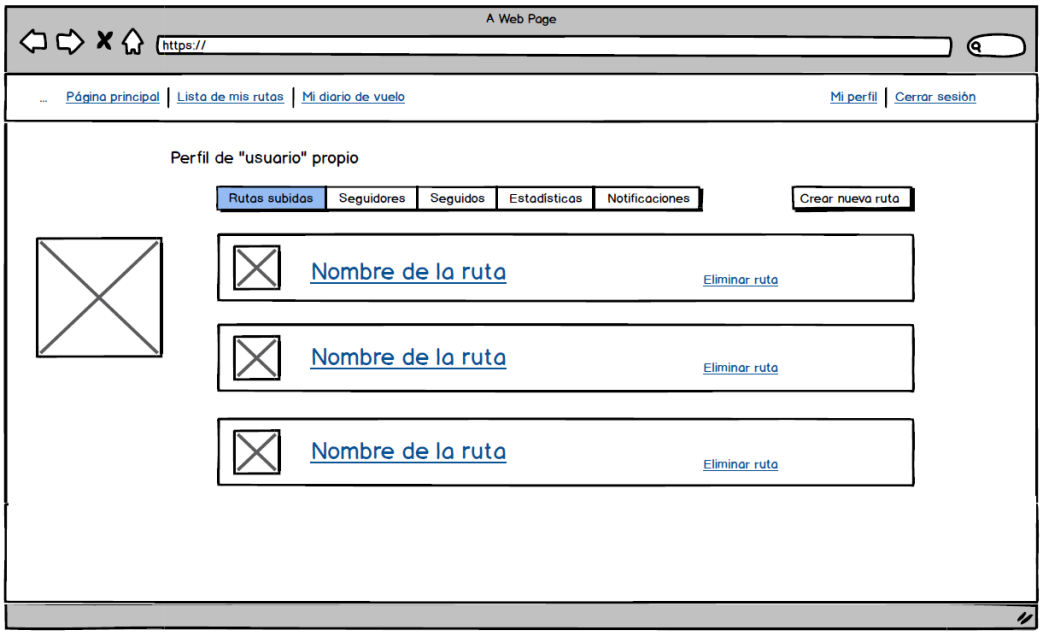


Figura B.6: Pantalla de lista de rutas en perfil propio.

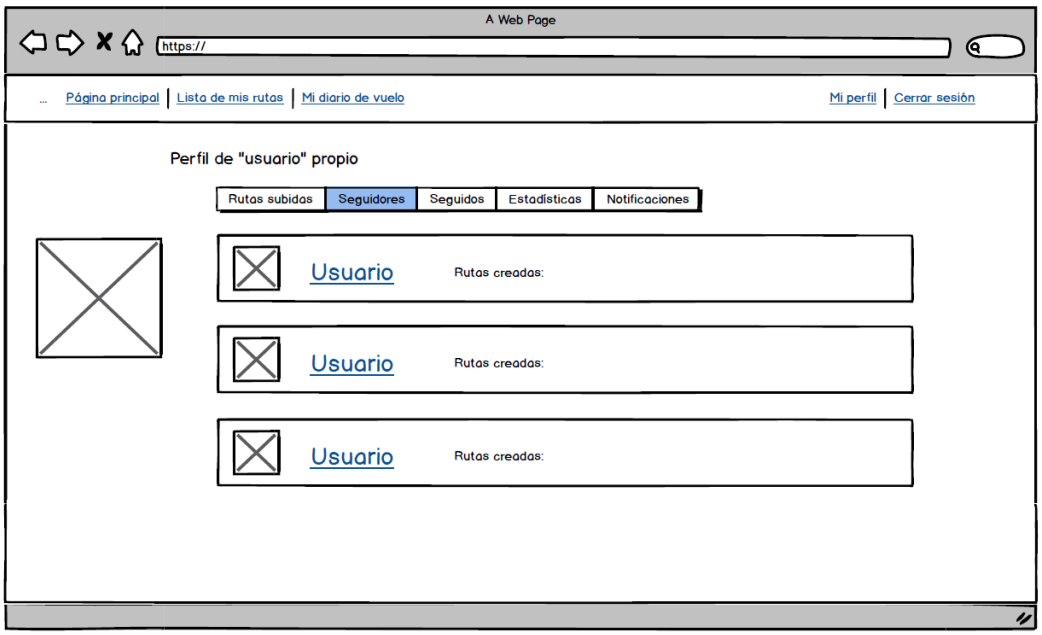


Figura B.7: Pantalla de seguidores en perfil.

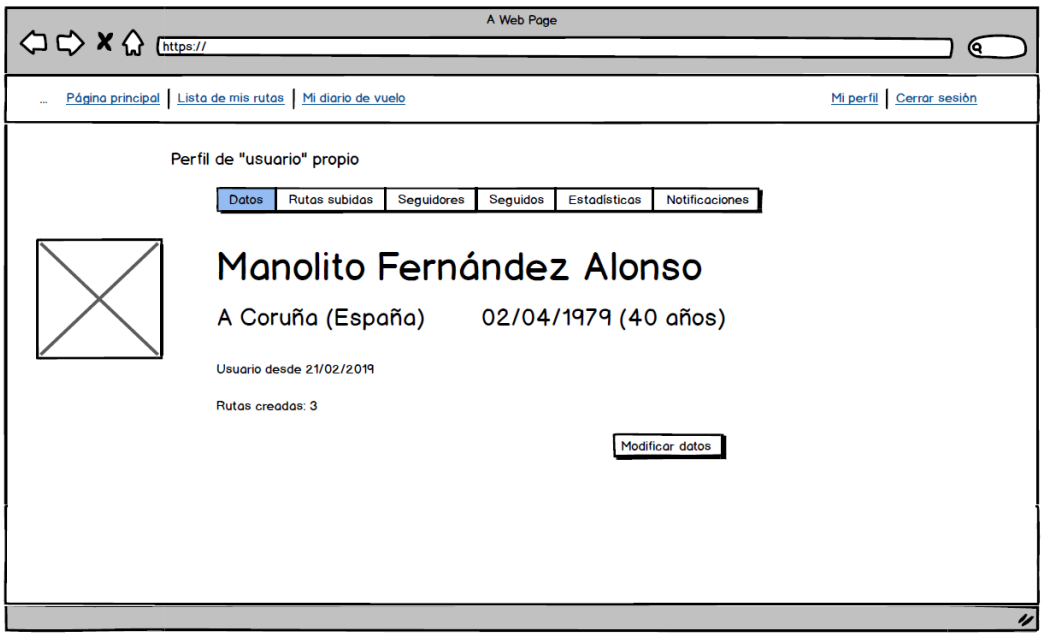


Figura B.8: Pantalla de información personal en perfil propio.

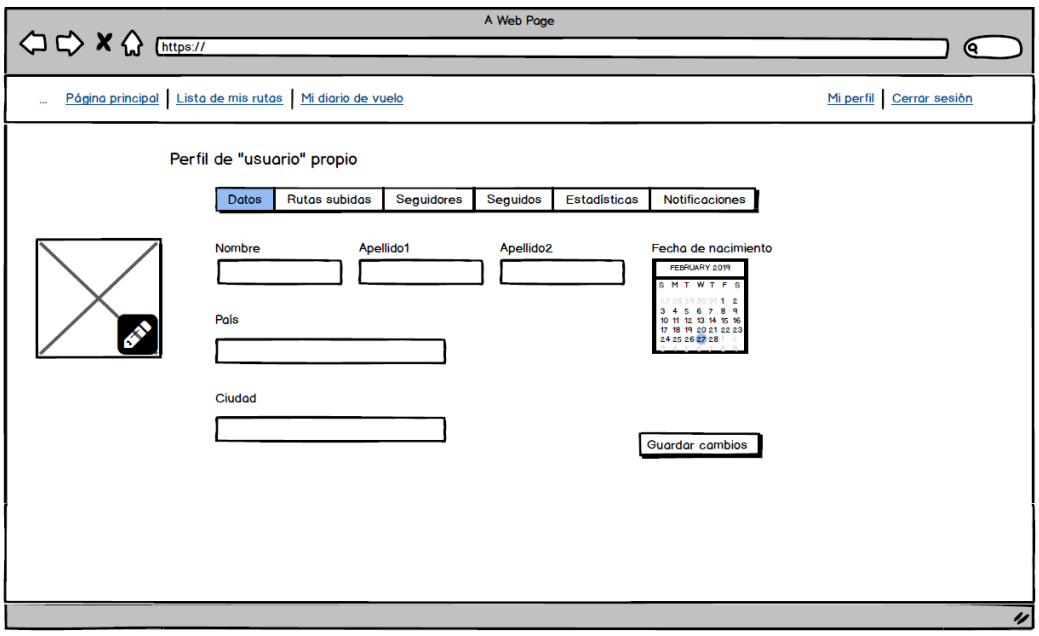


Figura B.9: Pantalla de edición de información personal.

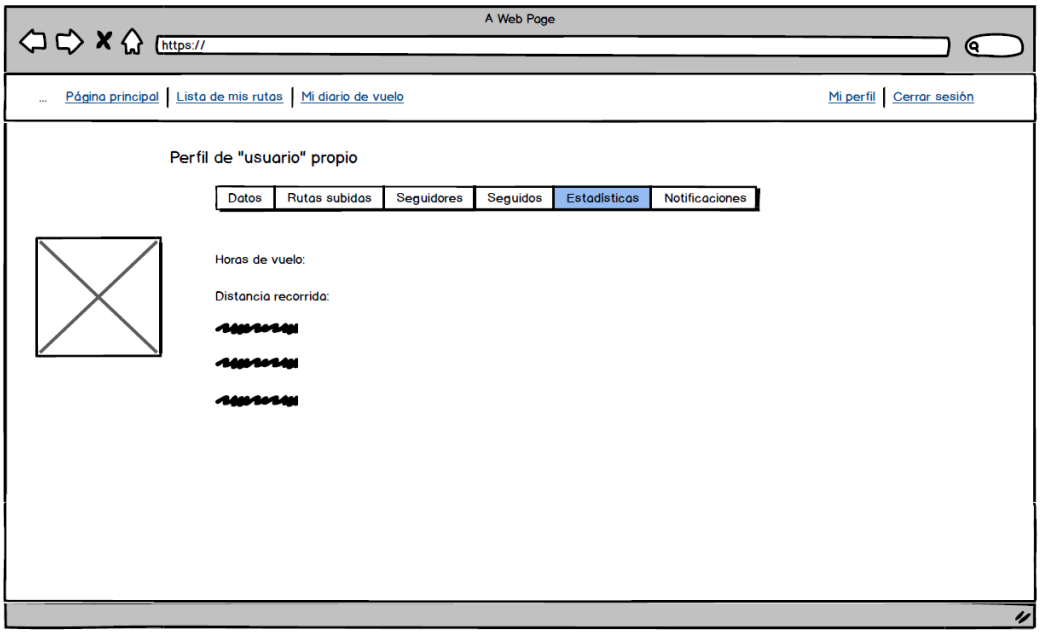


Figura B.10: Pantalla de estadísticas en perfil.

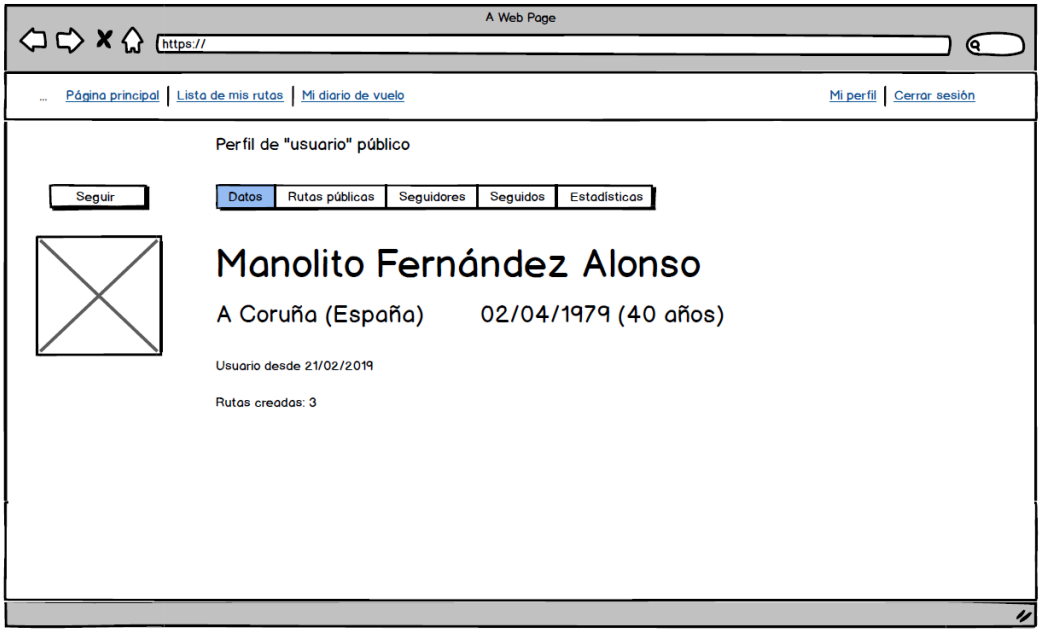


Figura B.11: Pantalla de información personal en perfil ajeno.

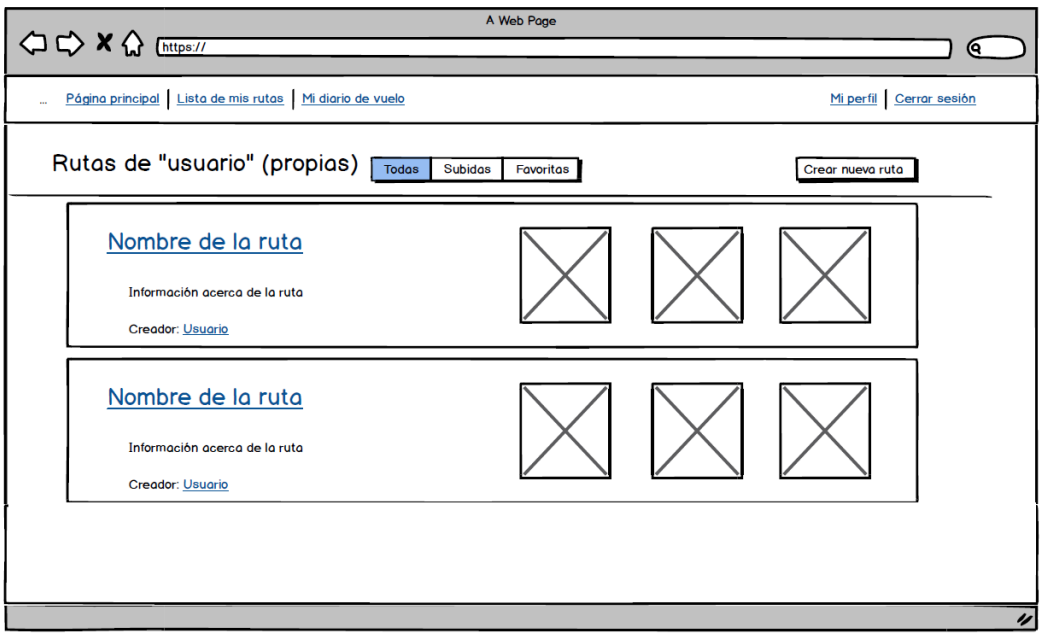


Figura B.12: Pantalla de listas de rutas para usuario registrado.

Figura B.13: Pantalla de detalle de ruta ajena.

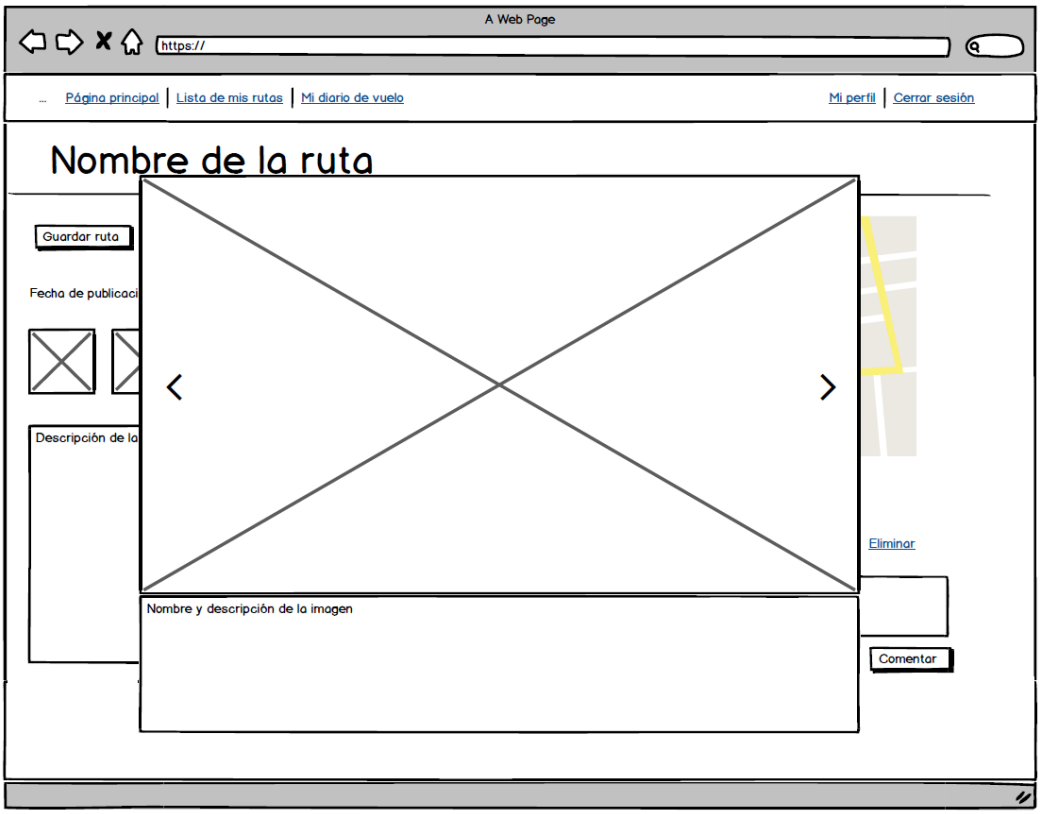


Figura B.14: Pantalla de detalle de imagen.

Figura B.15: Pantalla de detalle de ruta propia.

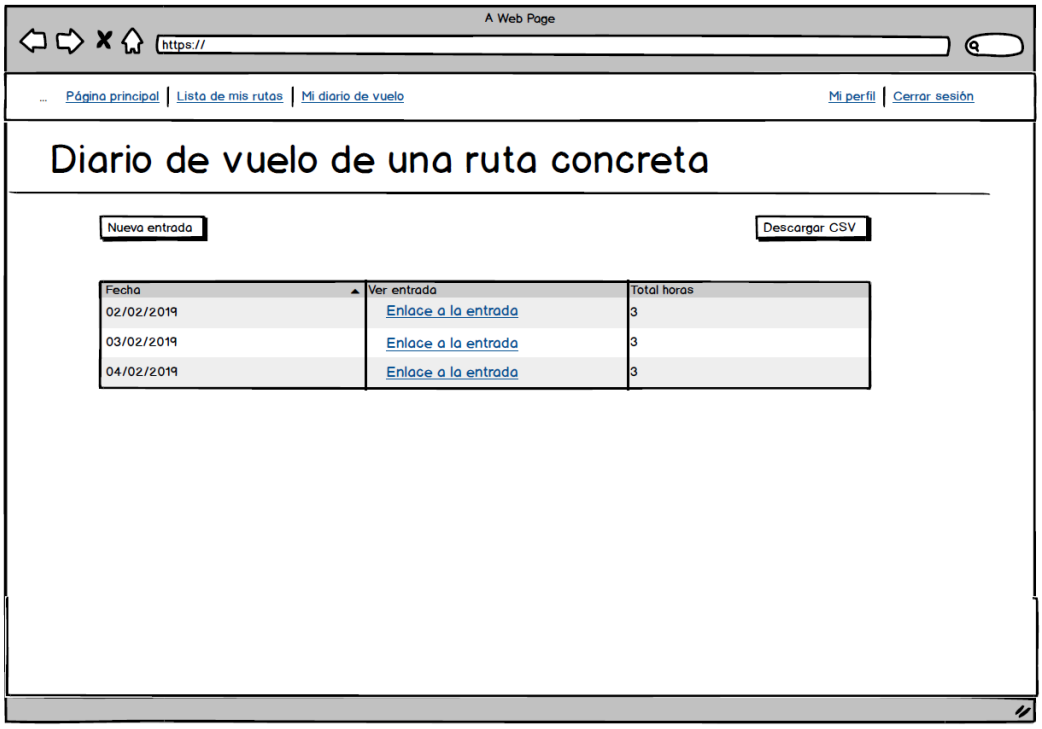


Figura B.16: Pantalla de diario de vuelo de ruta concreta.

Figura B.17: Pantalla de formulario de creación y edición de ruta.

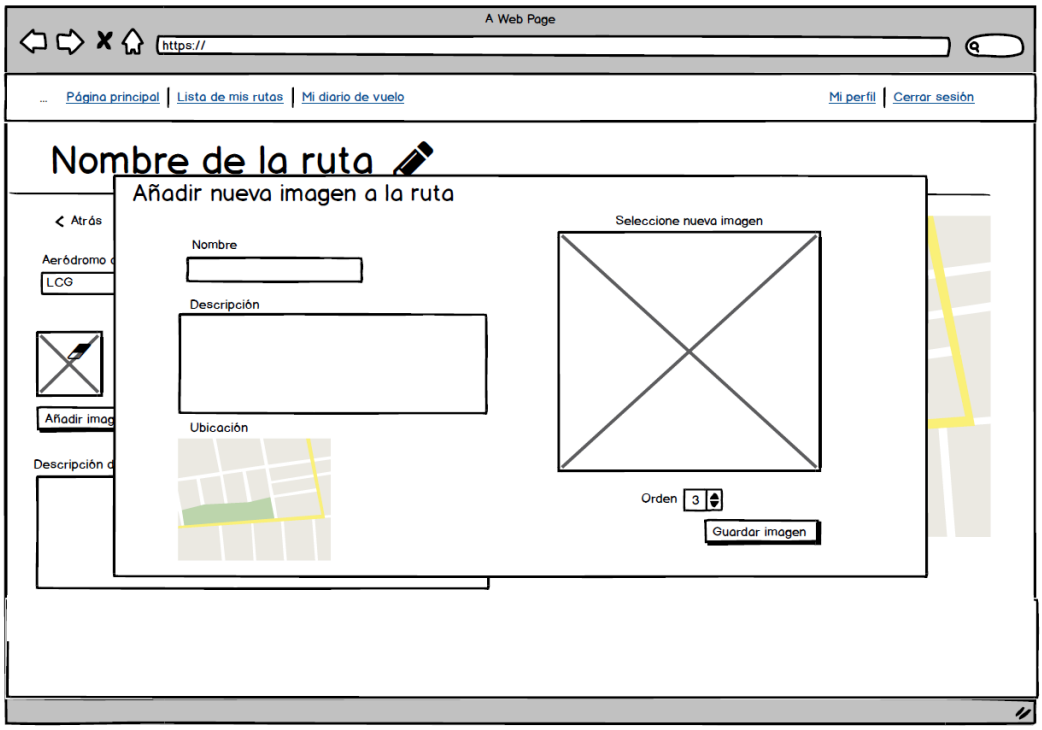


Figura B.18: Pantalla de creación de imagen asociada a ruta.

Figura B.19: Pantalla de diario de vuelo.

Figura B.20: Pantalla de creación y edición de vuelos.

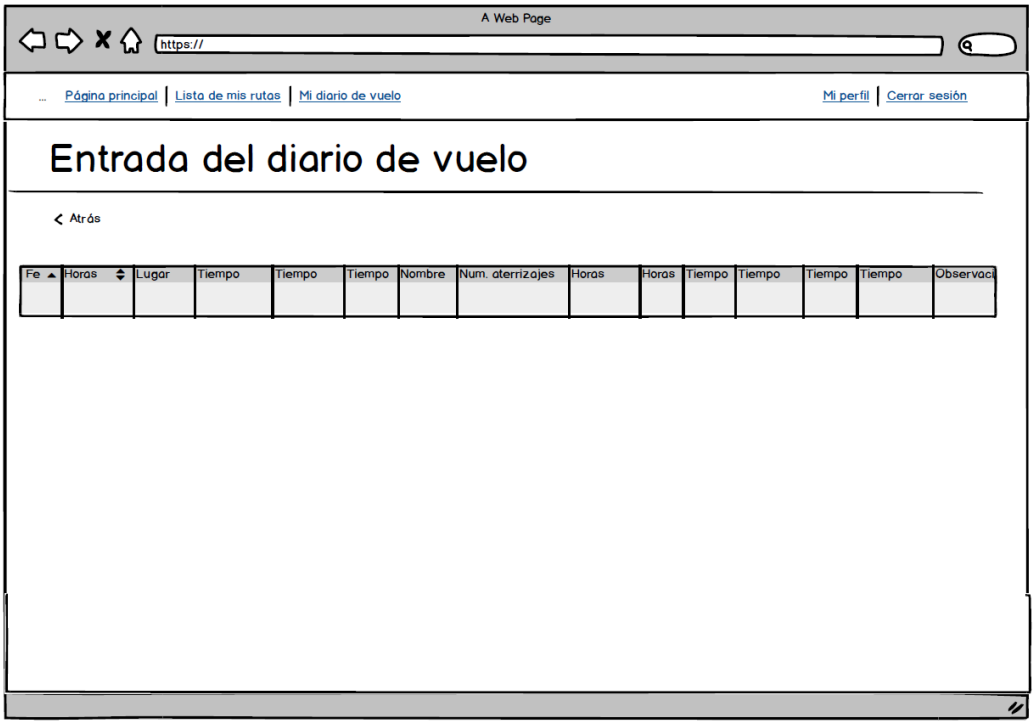


Figura B.21: Pantalla de detalle de vuelo.

Figura B.22: Pantalla principal para administradores.

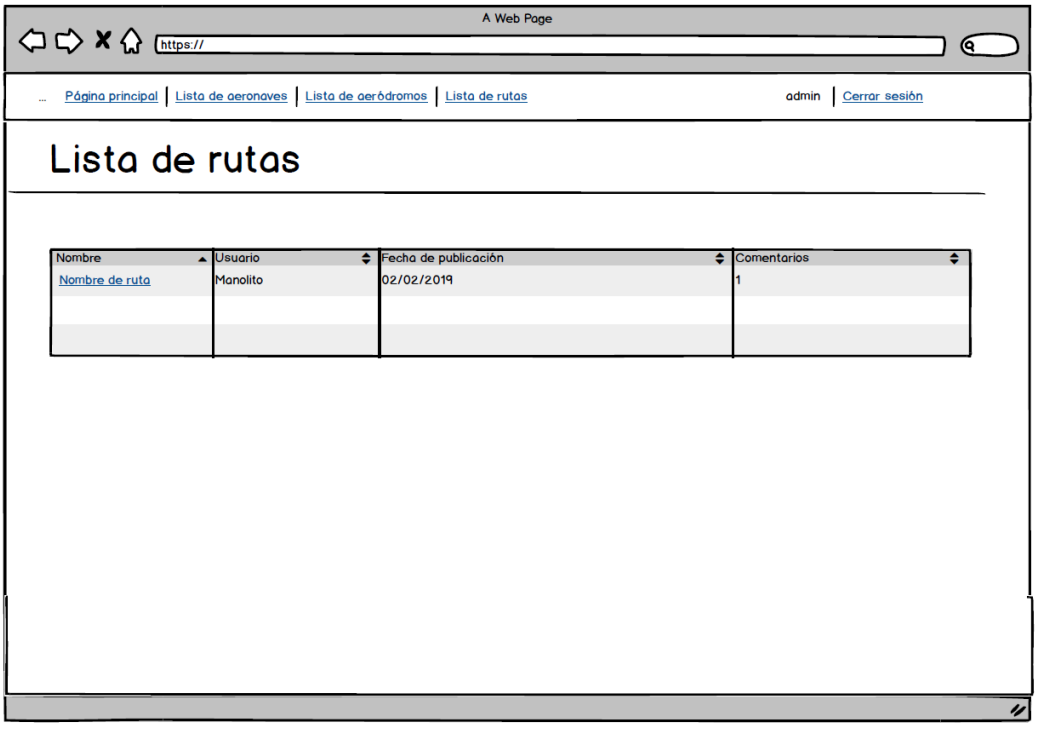


Figura B.23: Pantalla de lista de rutas para administradores.

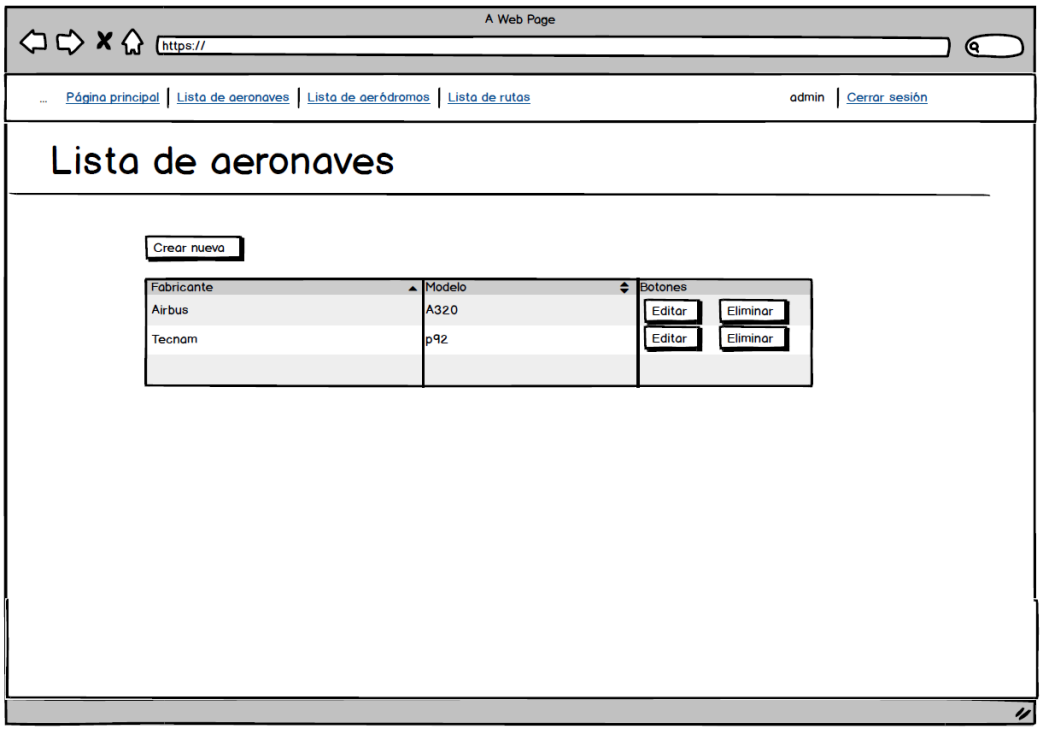


Figura B.24: Pantalla de lista de aeronaves.

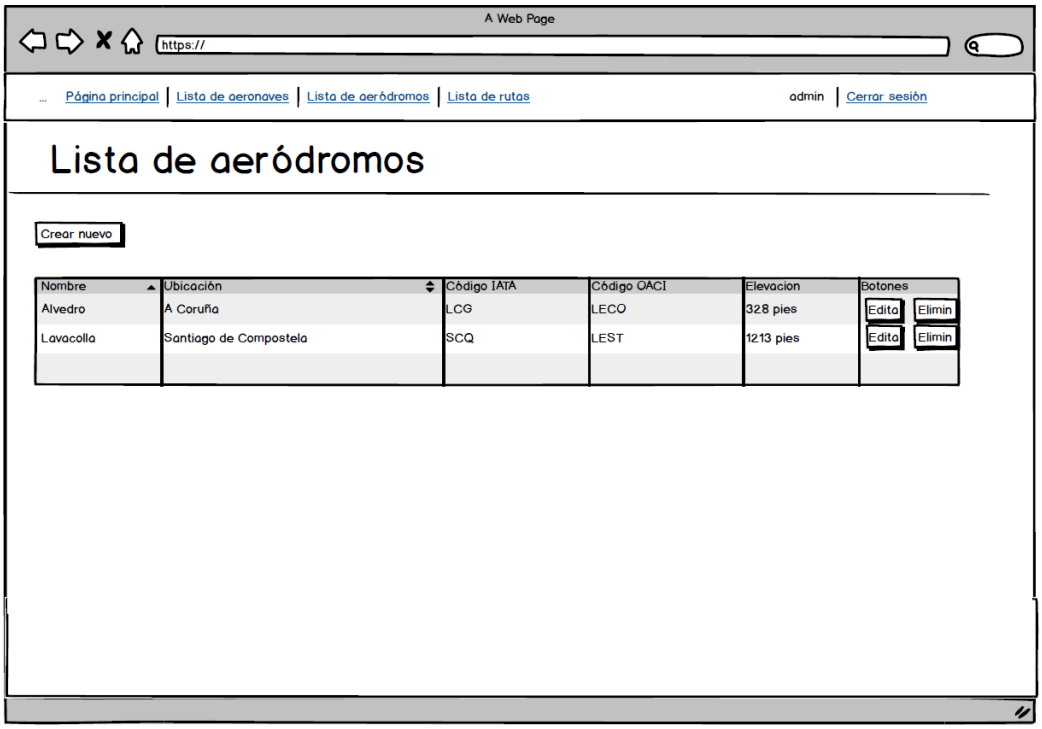


Figura B.25: Pantalla de lista de aeródromos.

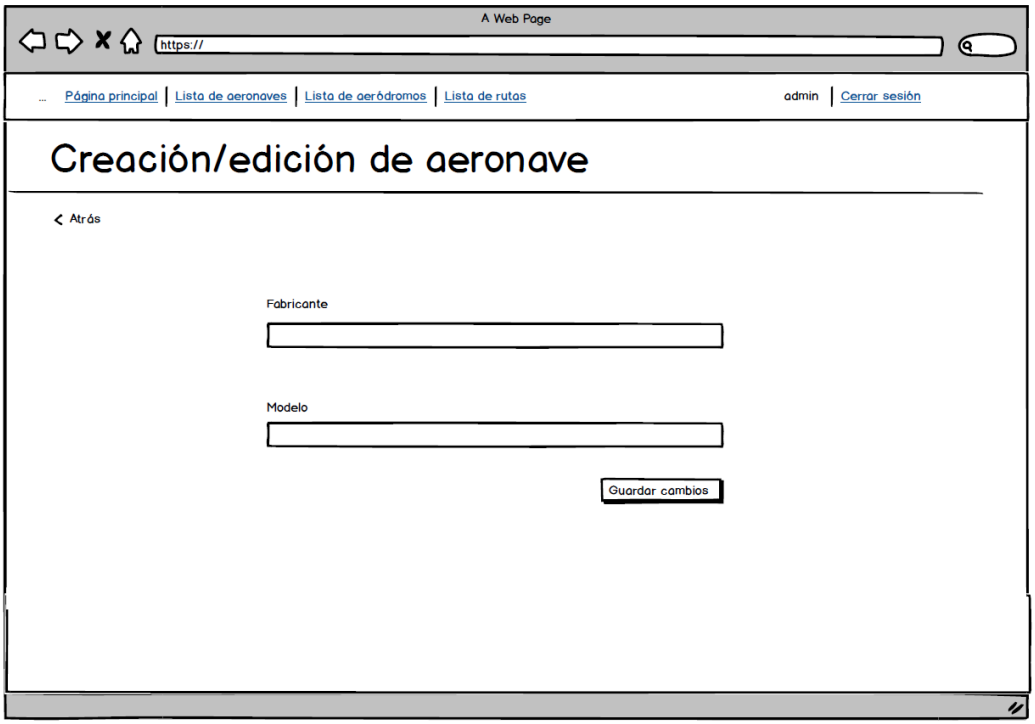


Figura B.26: Pantalla de creación y edición de aeronaves.

Figura B.27: Pantalla de creación y edición de aeródromos.

Apéndice C

Glosario de acrónimos

API *Application Programming Interface.*

CASE *Computer Aided Software Engineering.*

CRUD *Create, Read, Update and Delete.*

CSS *Cascading Style Sheets.*

CSV *Comma-Separated Values.*

DAO *Data Access Object.*

DOM *Document Object Model.*

DTO *Data Transfer Object.*

ES6 *ECMAScript v6.*

GPS *Global Positioning System.*

GPX *GPS eXchange Format.*

HQL *Hibernate Query Language.*

HTML *HyperText Markup Language.*

HTTP *Hypertext Transfer Protocol.*

IATA *International Air Transport Association.*

IDE *Integrated Development Environment.*

JWT *JSON Web Token.*

MVC *Model–View–Controller.*

OACI *Organización de Aviación Civil Internacional.*

ORM *Object-Relational Mapping.*

REST *Representational State Transfer.*

SGBD *Sistema de Gestión de Bases de Datos.*

SQL *Structured Query Language.*

UML *Unified Modeling Language.*

URL *Uniform Resource Locator.*

XML *eXtensible Markup Language .*